МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №9

Специальность ПО5

Выполнил:
А.А. Игнатюк,
студент группы ПО-5

Проверил:
А.А. Крощенко,
ст. преп. кафедры ИИТ,
«___» _________ 2022 г.

Брест 2022

Цель работы: Приобрести практические навыки разработки баз данных и начальной интеграции БД с кодом Java с помощью JDBC.

Вариант 5.

Задание.

Реализовать базу данных из не менее 5 таблиц на заданную тематику. При реализации продумать типизацию полей и внешние ключи в таблицах.

Визуализировать разработанную БД с помощью схемы, на которой отображены все таблицы и связи между ними.

На языке Java с использованием JDBC реализовать подключение к БД и выполнить основные типы запросов, продемонстрировать результаты преподавателю и включить тексты составленных запросов в отчет.

Основные типы запросов:

1. На выборку/на выборку с упорядочиванием (SELECT);
2. На добавление (INSERT INTO);
3. На удаление (DELETE FROM);
4. На модификацию (UPDATE).

Базу данные можно реализовать в любой СУБД (MySQL, PostgreSQL, SQLite и др.).

5) База данных “Сборка компьютера”.

Спецификация ввода: -

Спецификация вывода: содержимое базы данных в процессе работы программы.

Схема таблиц базы данных:

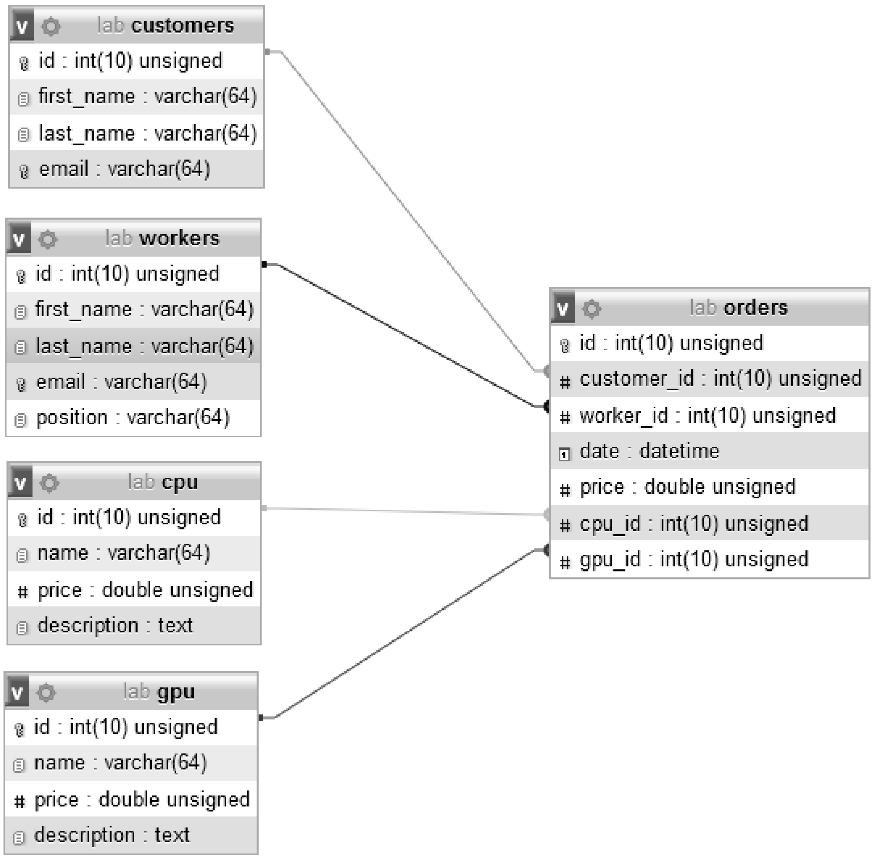
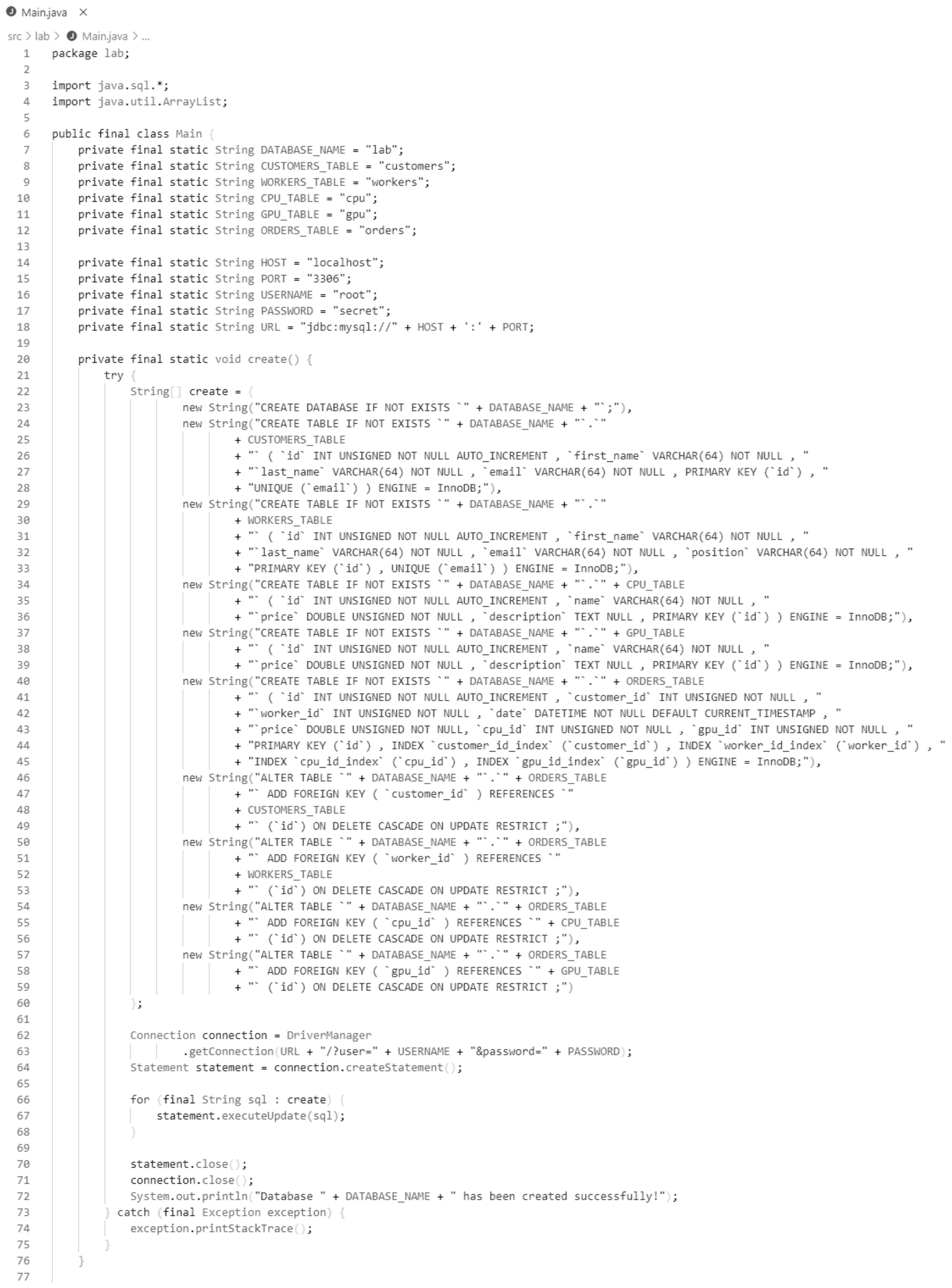


Рисунок 1 - Схема таблиц базы данных.

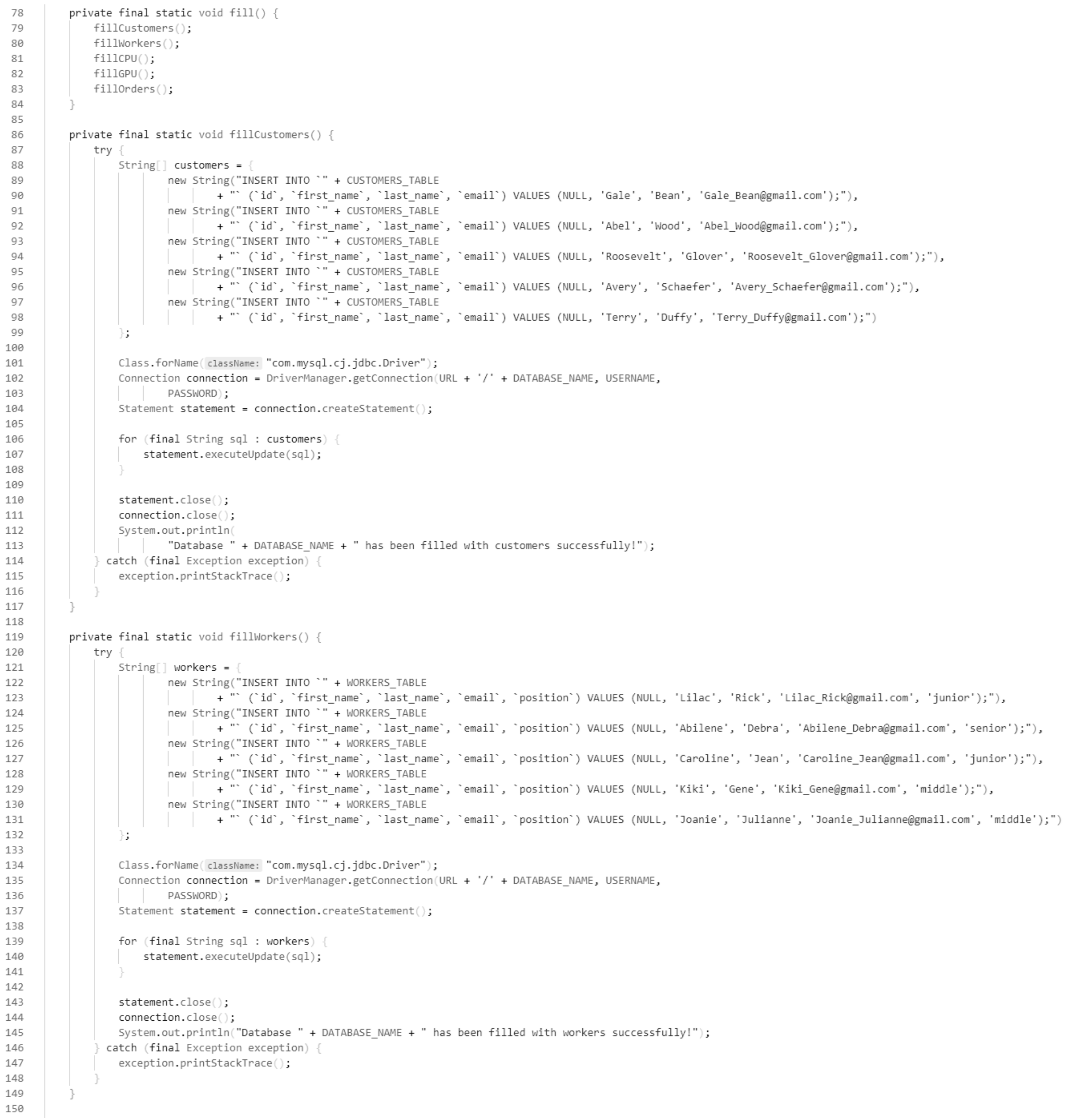
Код программы и результаты тестирования:



```java
package lab;

import java.sql.*;
import java.util.ArrayList;

public final class Main {
    private final static String DATABASE_NAME = "lab";
    private final static String CUSTOMERS_TABLE = "customers";
    private final static String WORKERS_TABLE = "workers";
    private final static String CPU_TABLE = "cpu";
    private final static String GPU_TABLE = "gpu";
    private final static String ORDERS_TABLE = "orders";

    private final static String HOST = "localhost";
    private final static String PORT = "3306";
    private final static String USERNAME = "root";
    private final static String PASSWORD = "secret";
    private final static String URL = "jdbc:mysql://" + HOST + ':' + PORT;

    private final static void create() {
        try {
            String[] create = {
                    new String("CREATE DATABASE IF NOT EXISTS `" + DATABASE_NAME + "`;"),
                    new String("CREATE TABLE IF NOT EXISTS `" + DATABASE_NAME + "`.`"
                            + CUSTOMERS_TABLE
                            + "` ( `id` INT UNSIGNED NOT NULL AUTO_INCREMENT , `first_name` VARCHAR(64) NOT NULL , "
                            + "`last_name` VARCHAR(64) NOT NULL , `email` VARCHAR(64) NOT NULL , PRIMARY KEY (`id`) , "
                            + "UNIQUE (`email`) ) ENGINE = InnoDB;"),
                    new String("CREATE TABLE IF NOT EXISTS `" + DATABASE_NAME + "`.`"
                            + WORKERS_TABLE
                            + "` ( `id` INT UNSIGNED NOT NULL AUTO_INCREMENT , `first_name` VARCHAR(64) NOT NULL , "
                            + "`last_name` VARCHAR(64) NOT NULL , `email` VARCHAR(64) NOT NULL , `position` VARCHAR(64) NOT NULL , "
                            + "PRIMARY KEY (`id`) , UNIQUE (`email`) ) ENGINE = InnoDB;"),
                    new String("CREATE TABLE IF NOT EXISTS `" + DATABASE_NAME + "`.`" + CPU_TABLE
                            + "` ( `id` INT UNSIGNED NOT NULL AUTO_INCREMENT , `name` VARCHAR(64) NOT NULL , "
                            + "`price` DOUBLE UNSIGNED NOT NULL , `description` TEXT NULL , PRIMARY KEY (`id`) ) ENGINE = InnoDB;"),
                    new String("CREATE TABLE IF NOT EXISTS `" + DATABASE_NAME + "`.`" + GPU_TABLE
                            + "` ( `id` INT UNSIGNED NOT NULL AUTO_INCREMENT , `name` VARCHAR(64) NOT NULL , "
                            + "`price` DOUBLE UNSIGNED NOT NULL , `description` TEXT NULL , PRIMARY KEY (`id`) ) ENGINE = InnoDB;"),
                    new String("CREATE TABLE IF NOT EXISTS `" + DATABASE_NAME + "`.`" + ORDERS_TABLE
                            + "` ( `id` INT UNSIGNED NOT NULL AUTO_INCREMENT , `customer_id` INT UNSIGNED NOT NULL , "
                            + "`worker_id` INT UNSIGNED NOT NULL , `date` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP , "
                            + "`price` DOUBLE UNSIGNED NOT NULL, `cpu_id` INT UNSIGNED NOT NULL , `gpu_id` INT UNSIGNED NOT NULL , "
                            + "PRIMARY KEY (`id`) , INDEX `customer_id_index` (`customer_id`) , INDEX `worker_id_index` (`worker_id`) , "
                            + "INDEX `cpu_id_index` (`cpu_id`) , INDEX `gpu_id_index` (`gpu_id`) ) ENGINE = InnoDB;"),
                    new String("ALTER TABLE `" + DATABASE_NAME + "`.`" + ORDERS_TABLE
                            + "` ADD FOREIGN KEY ( `customer_id` ) REFERENCES `"
                            + CUSTOMERS_TABLE
                            + "` (`id`) ON DELETE CASCADE ON UPDATE RESTRICT ;"),
                    new String("ALTER TABLE `" + DATABASE_NAME + "`.`" + ORDERS_TABLE
                            + "` ADD FOREIGN KEY ( `worker_id` ) REFERENCES `"
                            + WORKERS_TABLE
                            + "` (`id`) ON DELETE CASCADE ON UPDATE RESTRICT ;"),
                    new String("ALTER TABLE `" + DATABASE_NAME + "`.`" + ORDERS_TABLE
                            + "` ADD FOREIGN KEY ( `cpu_id` ) REFERENCES `" + CPU_TABLE
                            + "` (`id`) ON DELETE CASCADE ON UPDATE RESTRICT ;"),
                    new String("ALTER TABLE `" + DATABASE_NAME + "`.`" + ORDERS_TABLE
                            + "` ADD FOREIGN KEY ( `gpu_id` ) REFERENCES `" + GPU_TABLE
                            + "` (`id`) ON DELETE CASCADE ON UPDATE RESTRICT ;")
            };

            Connection connection = DriverManager
                    .getConnection(URL + "/?user=" + USERNAME + "&password=" + PASSWORD);
            Statement statement = connection.createStatement();

            for (final String sql : create) {
                statement.executeUpdate(sql);
            }

            statement.close();
            connection.close();
            System.out.println("Database " + DATABASE_NAME + " has been created successfully!");
        } catch (final Exception exception) {
            exception.printStackTrace();
        }
    }

```

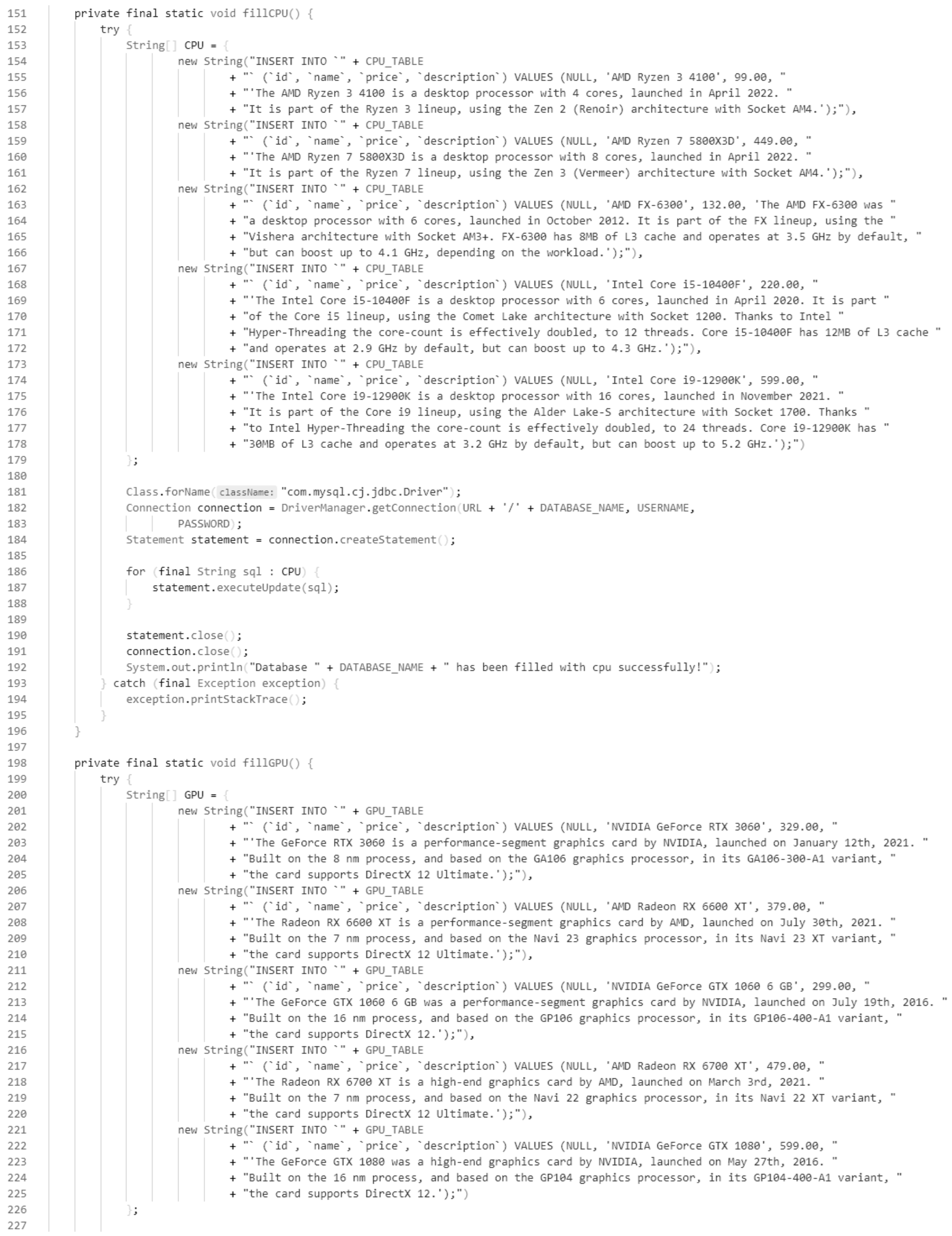
Рисунок 2 - Исходный код программы.

Продолжение рисунка 2.



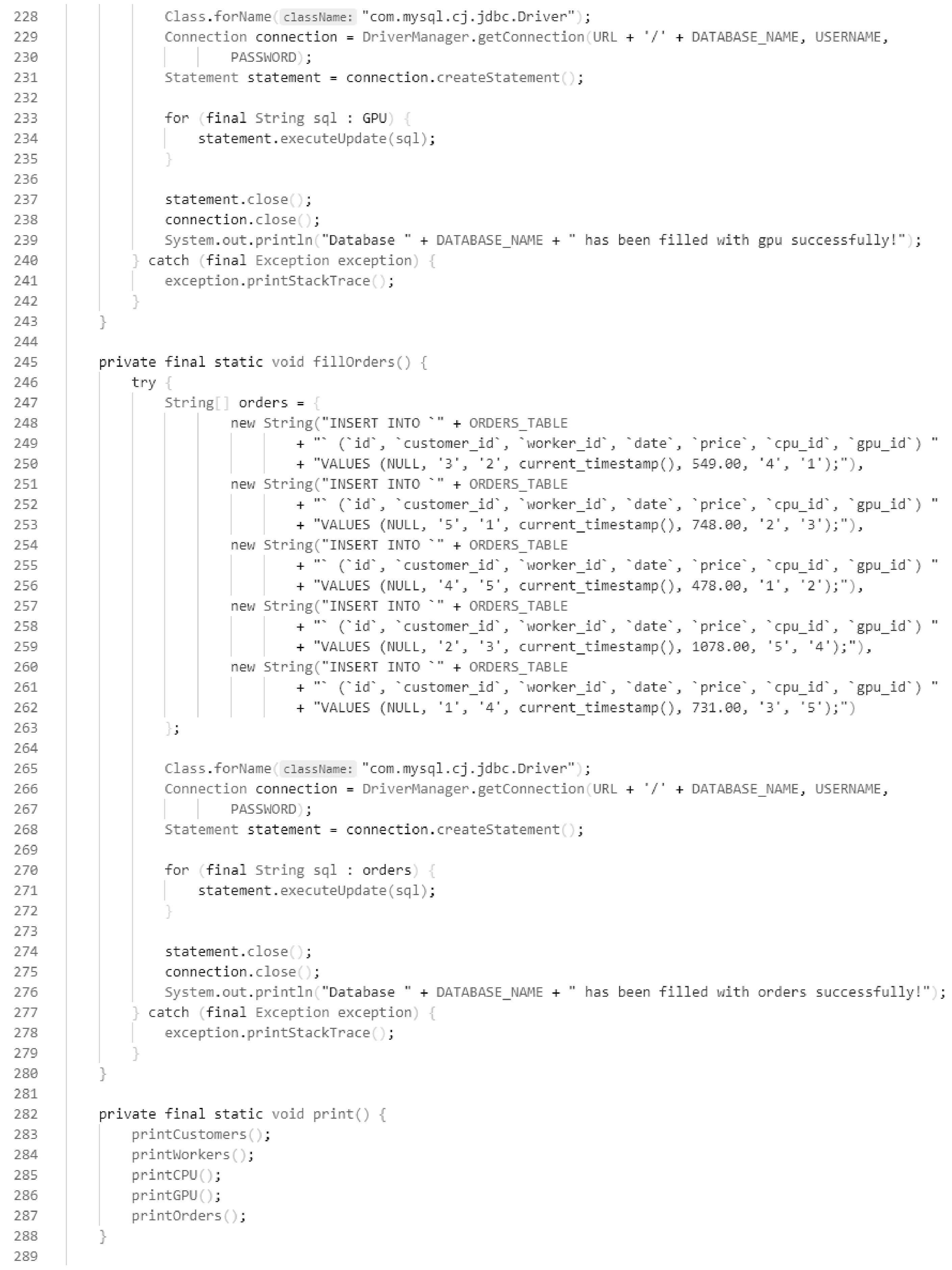
```java
private final static void fill() {
    fillCustomers();
    fillWorkers();
    fillCPU();
    fillGPU();
    fillOrders();
}

private final static void fillCustomers() {
    try {
        String[] customers = {
                new String("INSERT INTO `" + CUSTOMERS_TABLE
                        + "` (`id`, `first_name`, `last_name`, `email`) VALUES (NULL, 'Gale', 'Bean', 'Gale_Bean@gmail.com');"),
                new String("INSERT INTO `" + CUSTOMERS_TABLE
                        + "` (`id`, `first_name`, `last_name`, `email`) VALUES (NULL, 'Abel', 'Wood', 'Abel_Wood@gmail.com');"),
                new String("INSERT INTO `" + CUSTOMERS_TABLE
                        + "` (`id`, `first_name`, `last_name`, `email`) VALUES (NULL, 'Roosevelt', 'Glover', 'Roosevelt_Glover@gmail.com');"),
                new String("INSERT INTO `" + CUSTOMERS_TABLE
                        + "` (`id`, `first_name`, `last_name`, `email`) VALUES (NULL, 'Avery', 'Schaefer', 'Avery_Schaefer@gmail.com');"),
                new String("INSERT INTO `" + CUSTOMERS_TABLE
                        + "` (`id`, `first_name`, `last_name`, `email`) VALUES (NULL, 'Terry', 'Duffy', 'Terry_Duffy@gmail.com');")
        };

        Class.forName( className: "com.mysql.cj.jdbc.Driver");
        Connection connection = DriverManager.getConnection(URL + '/' + DATABASE_NAME, USERNAME,
                PASSWORD);
        Statement statement = connection.createStatement();

        for (final String sql : customers) {
            statement.executeUpdate(sql);
        }

        statement.close();
        connection.close();
        System.out.println(
                "Database " + DATABASE_NAME + " has been filled with customers successfully!");
    } catch (final Exception exception) {
        exception.printStackTrace();
    }
}

private final static void fillWorkers() {
    try {
        String[] workers = {
                new String("INSERT INTO `" + WORKERS_TABLE
                        + "` (`id`, `first_name`, `last_name`, `email`, `position`) VALUES (NULL, 'Lilac', 'Rick', 'Lilac_Rick@gmail.com', 'junior');"),
                new String("INSERT INTO `" + WORKERS_TABLE
                        + "` (`id`, `first_name`, `last_name`, `email`, `position`) VALUES (NULL, 'Abilene', 'Debra', 'Abilene_Debra@gmail.com', 'senior');"),
                new String("INSERT INTO `" + WORKERS_TABLE
                        + "` (`id`, `first_name`, `last_name`, `email`, `position`) VALUES (NULL, 'Caroline', 'Jean', 'Caroline_Jean@gmail.com', 'junior');"),
                new String("INSERT INTO `" + WORKERS_TABLE
                        + "` (`id`, `first_name`, `last_name`, `email`, `position`) VALUES (NULL, 'Kiki', 'Gene', 'Kiki_Gene@gmail.com', 'middle');"),
                new String("INSERT INTO `" + WORKERS_TABLE
                        + "` (`id`, `first_name`, `last_name`, `email`, `position`) VALUES (NULL, 'Joanie', 'Julianne', 'Joanie_Julianne@gmail.com', 'middle');")
        };

        Class.forName( className: "com.mysql.cj.jdbc.Driver");
        Connection connection = DriverManager.getConnection(URL + '/' + DATABASE_NAME, USERNAME,
                PASSWORD);
        Statement statement = connection.createStatement();

        for (final String sql : workers) {
            statement.executeUpdate(sql);
        }

        statement.close();
        connection.close();
        System.out.println("Database " + DATABASE_NAME + " has been filled with workers successfully!");
    } catch (final Exception exception) {
        exception.printStackTrace();
    }
}
```

Продолжение рисунка 2.



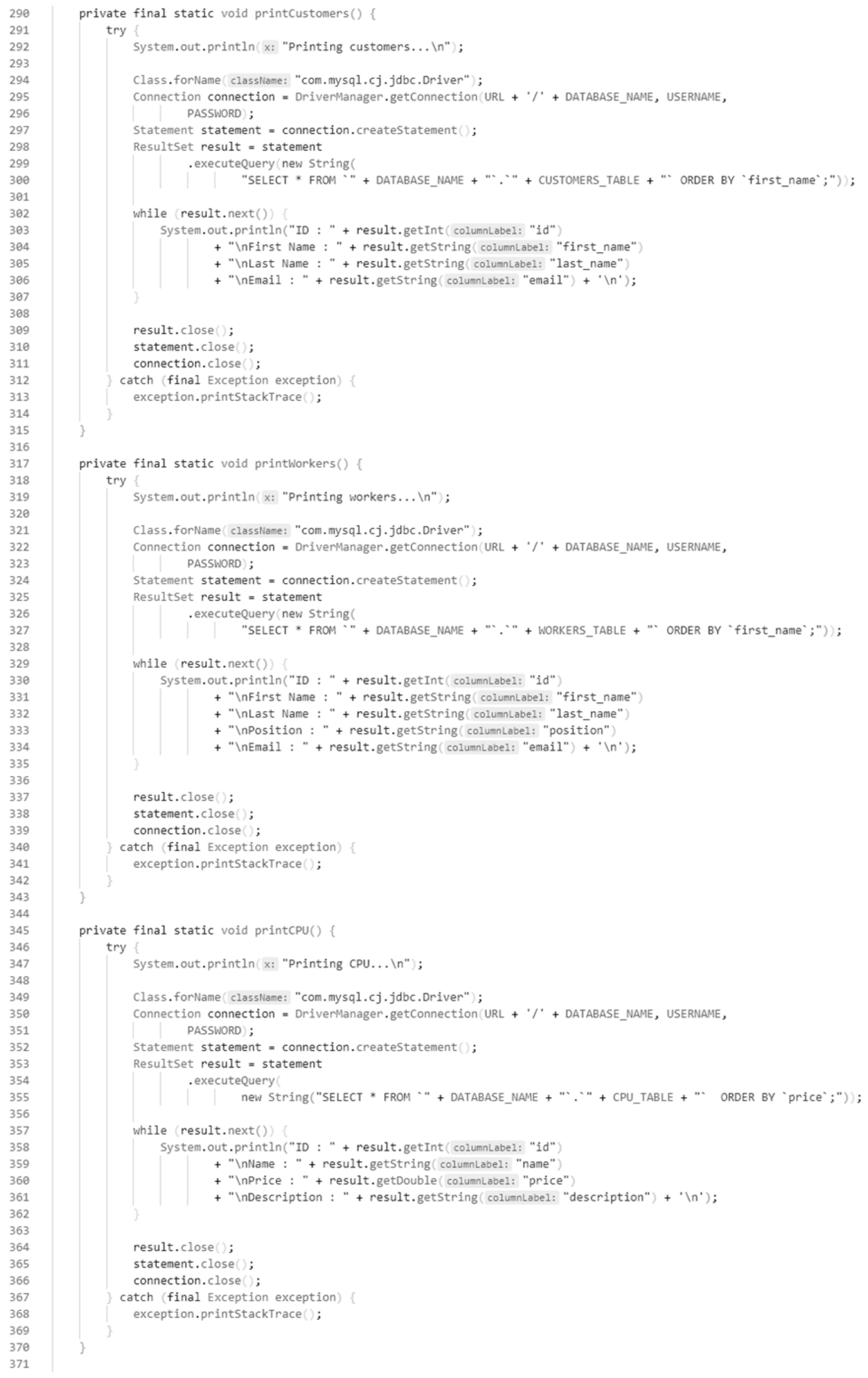
```java
    private final static void fillCPU() {
        try {
            String[] CPU = {
                    new String("INSERT INTO `" + CPU_TABLE
                            + "` (`id`, `name`, `price`, `description`) VALUES (NULL, 'AMD Ryzen 3 4100', 99.00, "
                            + "'The AMD Ryzen 3 4100 is a desktop processor with 4 cores, launched in April 2022. "
                            + "It is part of the Ryzen 3 lineup, using the Zen 2 (Renoir) architecture with Socket AM4.');"),
                    new String("INSERT INTO `" + CPU_TABLE
                            + "` (`id`, `name`, `price`, `description`) VALUES (NULL, 'AMD Ryzen 7 5800X3D', 449.00, "
                            + "'The AMD Ryzen 7 5800X3D is a desktop processor with 8 cores, launched in April 2022. "
                            + "It is part of the Ryzen 7 lineup, using the Zen 3 (Vermeer) architecture with Socket AM4.');"),
                    new String("INSERT INTO `" + CPU_TABLE
                            + "` (`id`, `name`, `price`, `description`) VALUES (NULL, 'AMD FX-6300', 132.00, 'The AMD FX-6300 was "
                            + "a desktop processor with 6 cores, launched in October 2012. It is part of the FX lineup, using the "
                            + "Vishera architecture with Socket AM3+. FX-6300 has 8MB of L3 cache and operates at 3.5 GHz by default, "
                            + "but can boost up to 4.1 GHz, depending on the workload.');"),
                    new String("INSERT INTO `" + CPU_TABLE
                            + "` (`id`, `name`, `price`, `description`) VALUES (NULL, 'Intel Core i5-10400F', 220.00, "
                            + "'The Intel Core i5-10400F is a desktop processor with 6 cores, launched in April 2020. It is part "
                            + "of the Core i5 lineup, using the Comet Lake architecture with Socket 1200. Thanks to Intel "
                            + "Hyper-Threading the core-count is effectively doubled, to 12 threads. Core i5-10400F has 12MB of L3 cache "
                            + "and operates at 2.9 GHz by default, but can boost up to 4.3 GHz.');"),
                    new String("INSERT INTO `" + CPU_TABLE
                            + "` (`id`, `name`, `price`, `description`) VALUES (NULL, 'Intel Core i9-12900K', 599.00, "
                            + "'The Intel Core i9-12900K is a desktop processor with 16 cores, launched in November 2021. "
                            + "It is part of the Core i9 lineup, using the Alder Lake-S architecture with Socket 1700. Thanks "
                            + "to Intel Hyper-Threading the core-count is effectively doubled, to 24 threads. Core i9-12900K has "
                            + "30MB of L3 cache and operates at 3.2 GHz by default, but can boost up to 5.2 GHz.');")
            };

            Class.forName( className: "com.mysql.cj.jdbc.Driver");
            Connection connection = DriverManager.getConnection(URL + '/' + DATABASE_NAME, USERNAME,
                    PASSWORD);
            Statement statement = connection.createStatement();

            for (final String sql : CPU) {
                statement.executeUpdate(sql);
            }

            statement.close();
            connection.close();
            System.out.println("Database " + DATABASE_NAME + " has been filled with cpu successfully!");
        } catch (final Exception exception) {
            exception.printStackTrace();
        }
    }

    private final static void fillGPU() {
        try {
            String[] GPU = {
                    new String("INSERT INTO `" + GPU_TABLE
                            + "` (`id`, `name`, `price`, `description`) VALUES (NULL, 'NVIDIA GeForce RTX 3060', 329.00, "
                            + "'The GeForce RTX 3060 is a performance-segment graphics card by NVIDIA, launched on January 12th, 2021. "
                            + "Built on the 8 nm process, and based on the GA106 graphics processor, in its GA106-300-A1 variant, "
                            + "the card supports DirectX 12 Ultimate.');"),
                    new String("INSERT INTO `" + GPU_TABLE
                            + "` (`id`, `name`, `price`, `description`) VALUES (NULL, 'AMD Radeon RX 6600 XT', 379.00, "
                            + "'The Radeon RX 6600 XT is a performance-segment graphics card by AMD, launched on July 30th, 2021. "
                            + "Built on the 7 nm process, and based on the Navi 23 graphics processor, in its Navi 23 XT variant, "
                            + "the card supports DirectX 12 Ultimate.');"),
                    new String("INSERT INTO `" + GPU_TABLE
                            + "` (`id`, `name`, `price`, `description`) VALUES (NULL, 'NVIDIA GeForce GTX 1060 6 GB', 299.00, "
                            + "'The GeForce GTX 1060 6 GB was a performance-segment graphics card by NVIDIA, launched on July 19th, 2016. "
                            + "Built on the 16 nm process, and based on the GP106 graphics processor, in its GP106-400-A1 variant, "
                            + "the card supports DirectX 12.');"),
                    new String("INSERT INTO `" + GPU_TABLE
                            + "` (`id`, `name`, `price`, `description`) VALUES (NULL, 'AMD Radeon RX 6700 XT', 479.00, "
                            + "'The Radeon RX 6700 XT is a high-end graphics card by AMD, launched on March 3rd, 2021. "
                            + "Built on the 7 nm process, and based on the Navi 22 graphics processor, in its Navi 22 XT variant, "
                            + "the card supports DirectX 12 Ultimate.');"),
                    new String("INSERT INTO `" + GPU_TABLE
                            + "` (`id`, `name`, `price`, `description`) VALUES (NULL, 'NVIDIA GeForce GTX 1080', 599.00, "
                            + "'The GeForce GTX 1080 was a high-end graphics card by NVIDIA, launched on May 27th, 2016. "
                            + "Built on the 16 nm process, and based on the GP104 graphics processor, in its GP104-400-A1 variant, "
                            + "the card supports DirectX 12.');")
            };

```

Продолжение рисунка 2.



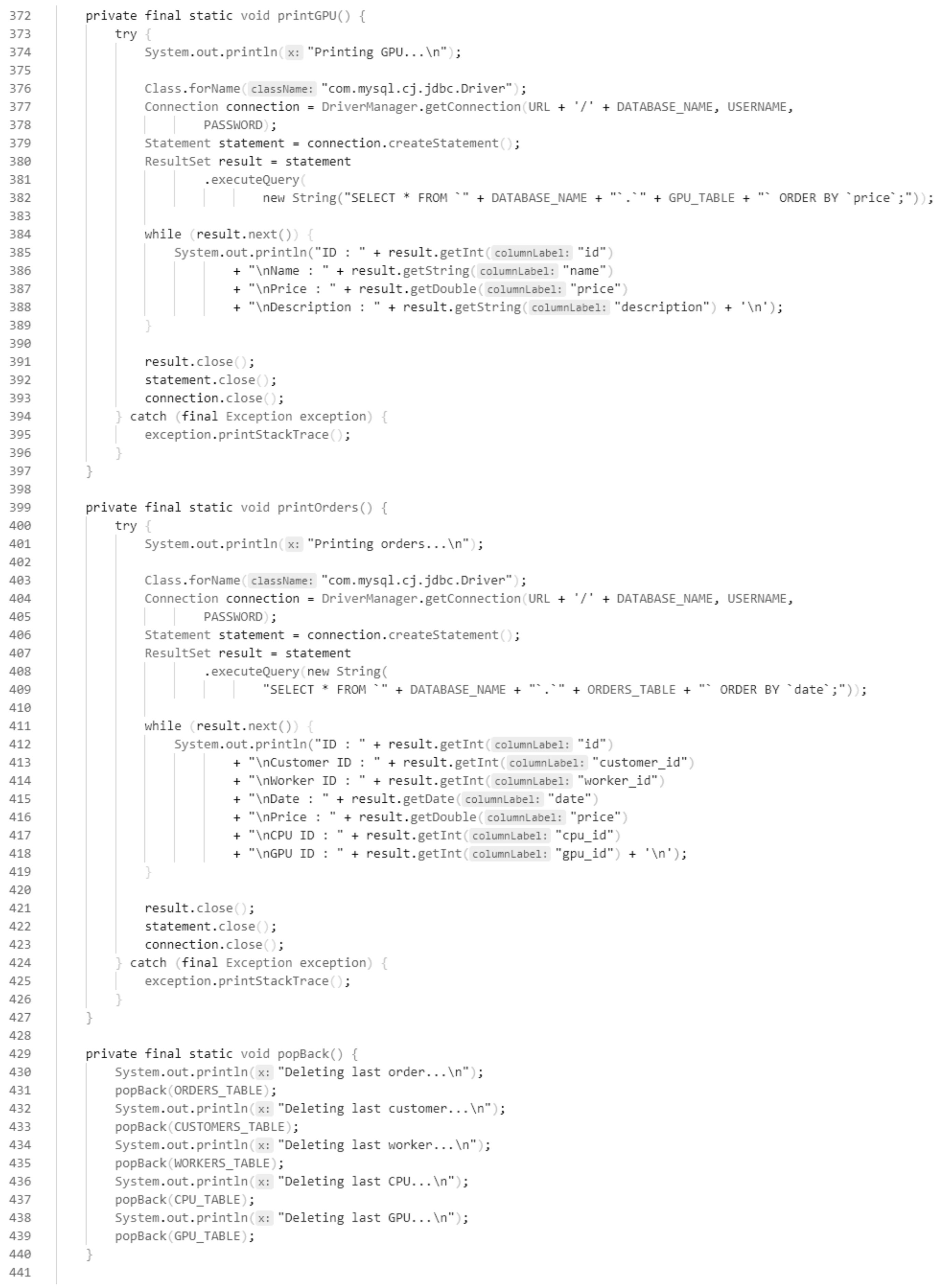
```java
            Class.forName(className: "com.mysql.cj.jdbc.Driver");
            Connection connection = DriverManager.getConnection(URL + '/' + DATABASE_NAME, USERNAME,
                    PASSWORD);
            Statement statement = connection.createStatement();

            for (final String sql : GPU) {
                statement.executeUpdate(sql);
            }

            statement.close();
            connection.close();
            System.out.println("Database " + DATABASE_NAME + " has been filled with gpu successfully!");
        } catch (final Exception exception) {
            exception.printStackTrace();
        }
    }

    private final static void fillOrders() {
        try {
            String[] orders = {
                    new String("INSERT INTO `" + ORDERS_TABLE
                            + "` (`id`, `customer_id`, `worker_id`, `date`, `price`, `cpu_id`, `gpu_id`) "
                            + "VALUES (NULL, '3', '2', current_timestamp(), 549.00, '4', '1');"),
                    new String("INSERT INTO `" + ORDERS_TABLE
                            + "` (`id`, `customer_id`, `worker_id`, `date`, `price`, `cpu_id`, `gpu_id`) "
                            + "VALUES (NULL, '5', '1', current_timestamp(), 748.00, '2', '3');"),
                    new String("INSERT INTO `" + ORDERS_TABLE
                            + "` (`id`, `customer_id`, `worker_id`, `date`, `price`, `cpu_id`, `gpu_id`) "
                            + "VALUES (NULL, '4', '5', current_timestamp(), 478.00, '1', '2');"),
                    new String("INSERT INTO `" + ORDERS_TABLE
                            + "` (`id`, `customer_id`, `worker_id`, `date`, `price`, `cpu_id`, `gpu_id`) "
                            + "VALUES (NULL, '2', '3', current_timestamp(), 1078.00, '5', '4');"),
                    new String("INSERT INTO `" + ORDERS_TABLE
                            + "` (`id`, `customer_id`, `worker_id`, `date`, `price`, `cpu_id`, `gpu_id`) "
                            + "VALUES (NULL, '1', '4', current_timestamp(), 731.00, '3', '5');")
            };

            Class.forName(className: "com.mysql.cj.jdbc.Driver");
            Connection connection = DriverManager.getConnection(URL + '/' + DATABASE_NAME, USERNAME,
                    PASSWORD);
            Statement statement = connection.createStatement();

            for (final String sql : orders) {
                statement.executeUpdate(sql);
            }

            statement.close();
            connection.close();
            System.out.println("Database " + DATABASE_NAME + " has been filled with orders successfully!");
        } catch (final Exception exception) {
            exception.printStackTrace();
        }
    }

    private final static void print() {
        printCustomers();
        printWorkers();
        printCPU();
        printGPU();
        printOrders();
    }

```

Продолжение рисунка 2.



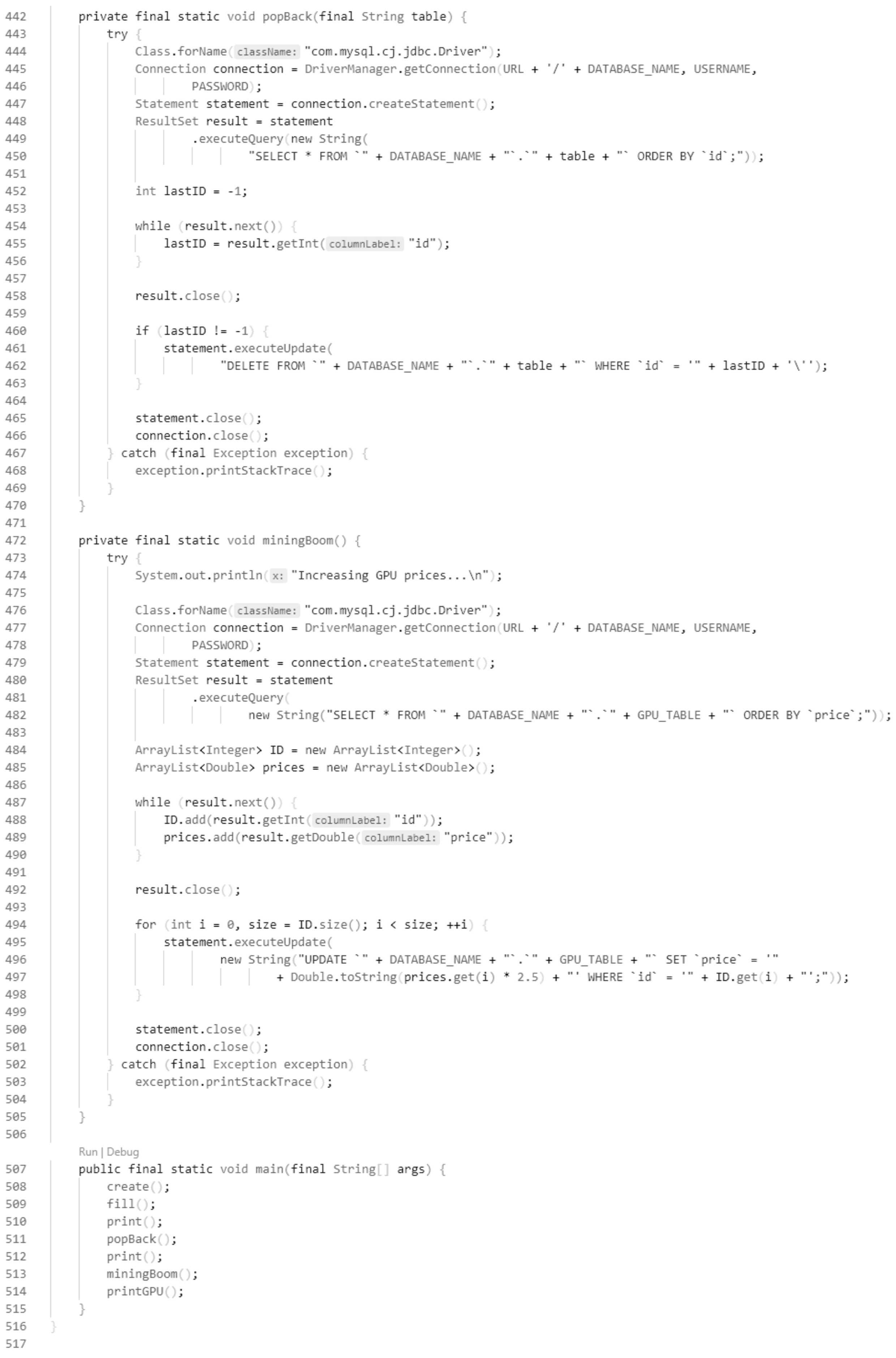
```
private final static void printCustomers() {
    try {
        System.out.println(x: "Printing customers...\n");

        Class.forName(className: "com.mysql.cj.jdbc.Driver");
        Connection connection = DriverManager.getConnection(URL + '/' + DATABASE_NAME, USERNAME,
                PASSWORD);
        Statement statement = connection.createStatement();
        ResultSet result = statement
                .executeQuery(new String(
                        "SELECT * FROM `" + DATABASE_NAME + "`.`" + CUSTOMERS_TABLE + "` ORDER BY `first_name`;"));

        while (result.next()) {
            System.out.println("ID : " + result.getInt(columnLabel: "id")
                    + "\nFirst Name : " + result.getString(columnLabel: "first_name")
                    + "\nLast Name : " + result.getString(columnLabel: "last_name")
                    + "\nEmail : " + result.getString(columnLabel: "email") + '\n');
        }

        result.close();
        statement.close();
        connection.close();
    } catch (final Exception exception) {
        exception.printStackTrace();
    }
}

private final static void printWorkers() {
    try {
        System.out.println(x: "Printing workers...\n");

        Class.forName(className: "com.mysql.cj.jdbc.Driver");
        Connection connection = DriverManager.getConnection(URL + '/' + DATABASE_NAME, USERNAME,
                PASSWORD);
        Statement statement = connection.createStatement();
        ResultSet result = statement
                .executeQuery(new String(
                        "SELECT * FROM `" + DATABASE_NAME + "`.`" + WORKERS_TABLE + "` ORDER BY `first_name`;"));

        while (result.next()) {
            System.out.println("ID : " + result.getInt(columnLabel: "id")
                    + "\nFirst Name : " + result.getString(columnLabel: "first_name")
                    + "\nLast Name : " + result.getString(columnLabel: "last_name")
                    + "\nPosition : " + result.getString(columnLabel: "position")
                    + "\nEmail : " + result.getString(columnLabel: "email") + '\n');
        }

        result.close();
        statement.close();
        connection.close();
    } catch (final Exception exception) {
        exception.printStackTrace();
    }
}

private final static void printCPU() {
    try {
        System.out.println(x: "Printing CPU...\n");

        Class.forName(className: "com.mysql.cj.jdbc.Driver");
        Connection connection = DriverManager.getConnection(URL + '/' + DATABASE_NAME, USERNAME,
                PASSWORD);
        Statement statement = connection.createStatement();
        ResultSet result = statement
                .executeQuery(
                        new String("SELECT * FROM `" + DATABASE_NAME + "`.`" + CPU_TABLE + "`  ORDER BY `price`;"));

        while (result.next()) {
            System.out.println("ID : " + result.getInt(columnLabel: "id")
                    + "\nName : " + result.getString(columnLabel: "name")
                    + "\nPrice : " + result.getDouble(columnLabel: "price")
                    + "\nDescription : " + result.getString(columnLabel: "description") + '\n');
        }

        result.close();
        statement.close();
        connection.close();
    } catch (final Exception exception) {
        exception.printStackTrace();
    }
}
```

Продолжение рисунка 2.



```java
    private final static void printGPU() {
        try {
            System.out.println(x: "Printing GPU...\n");

            Class.forName(className: "com.mysql.cj.jdbc.Driver");
            Connection connection = DriverManager.getConnection(URL + '/' + DATABASE_NAME, USERNAME,
                    PASSWORD);
            Statement statement = connection.createStatement();
            ResultSet result = statement
                    .executeQuery(
                            new String("SELECT * FROM `" + DATABASE_NAME + "`.`" + GPU_TABLE + "` ORDER BY `price`;"));

            while (result.next()) {
                System.out.println("ID : " + result.getInt(columnLabel: "id")
                        + "\nName : " + result.getString(columnLabel: "name")
                        + "\nPrice : " + result.getDouble(columnLabel: "price")
                        + "\nDescription : " + result.getString(columnLabel: "description") + '\n');
            }

            result.close();
            statement.close();
            connection.close();
        } catch (final Exception exception) {
            exception.printStackTrace();
        }
    }

    private final static void printOrders() {
        try {
            System.out.println(x: "Printing orders...\n");

            Class.forName(className: "com.mysql.cj.jdbc.Driver");
            Connection connection = DriverManager.getConnection(URL + '/' + DATABASE_NAME, USERNAME,
                    PASSWORD);
            Statement statement = connection.createStatement();
            ResultSet result = statement
                    .executeQuery(new String(
                            "SELECT * FROM `" + DATABASE_NAME + "`.`" + ORDERS_TABLE + "` ORDER BY `date`;"));

            while (result.next()) {
                System.out.println("ID : " + result.getInt(columnLabel: "id")
                        + "\nCustomer ID : " + result.getInt(columnLabel: "customer_id")
                        + "\nWorker ID : " + result.getInt(columnLabel: "worker_id")
                        + "\nDate : " + result.getDate(columnLabel: "date")
                        + "\nPrice : " + result.getDouble(columnLabel: "price")
                        + "\nCPU ID : " + result.getInt(columnLabel: "cpu_id")
                        + "\nGPU ID : " + result.getInt(columnLabel: "gpu_id") + '\n');
            }

            result.close();
            statement.close();
            connection.close();
        } catch (final Exception exception) {
            exception.printStackTrace();
        }
    }

    private final static void popBack() {
        System.out.println(x: "Deleting last order...\n");
        popBack(ORDERS_TABLE);
        System.out.println(x: "Deleting last customer...\n");
        popBack(CUSTOMERS_TABLE);
        System.out.println(x: "Deleting last worker...\n");
        popBack(WORKERS_TABLE);
        System.out.println(x: "Deleting last CPU...\n");
        popBack(CPU_TABLE);
        System.out.println(x: "Deleting last GPU...\n");
        popBack(GPU_TABLE);
    }

```

Продолжение рисунка 2.



```
    private final static void popBack(final String table) {
        try {
            Class.forName(className: "com.mysql.cj.jdbc.Driver");
            Connection connection = DriverManager.getConnection(URL + '/' + DATABASE_NAME, USERNAME,
                    PASSWORD);
            Statement statement = connection.createStatement();
            ResultSet result = statement
                    .executeQuery(new String(
                            "SELECT * FROM `" + DATABASE_NAME + "`.`" + table + "` ORDER BY `id`;"));

            int lastID = -1;

            while (result.next()) {
                lastID = result.getInt(columnLabel: "id");
            }

            result.close();

            if (lastID != -1) {
                statement.executeUpdate(
                        "DELETE FROM `" + DATABASE_NAME + "`.`" + table + "` WHERE `id` = '" + lastID + '\'');
            }

            statement.close();
            connection.close();
        } catch (final Exception exception) {
            exception.printStackTrace();
        }
    }

    private final static void miningBoom() {
        try {
            System.out.println(x: "Increasing GPU prices...\n");

            Class.forName(className: "com.mysql.cj.jdbc.Driver");
            Connection connection = DriverManager.getConnection(URL + '/' + DATABASE_NAME, USERNAME,
                    PASSWORD);
            Statement statement = connection.createStatement();
            ResultSet result = statement
                    .executeQuery(
                            new String("SELECT * FROM `" + DATABASE_NAME + "`.`" + GPU_TABLE + "` ORDER BY `price`;"));

            ArrayList<Integer> ID = new ArrayList<Integer>();
            ArrayList<Double> prices = new ArrayList<Double>();

            while (result.next()) {
                ID.add(result.getInt(columnLabel: "id"));
                prices.add(result.getDouble(columnLabel: "price"));
            }

            result.close();

            for (int i = 0, size = ID.size(); i < size; ++i) {
                statement.executeUpdate(
                        new String("UPDATE `" + DATABASE_NAME + "`.`" + GPU_TABLE + "` SET `price` = '"
                                + Double.toString(prices.get(i) * 2.5) + "' WHERE `id` = '" + ID.get(i) + "';"));
            }

            statement.close();
            connection.close();
        } catch (final Exception exception) {
            exception.printStackTrace();
        }
    }

    Run | Debug
    public final static void main(final String[] args) {
        create();
        fill();
        print();
        popBack();
        print();
        miningBoom();
        printGPU();
    }
}
```

Результат работы программыы:

```
PS D:\Documents\Visual Studio Code\Java\lab>  & 'C:\Program Files\Eclipse Foundation\jdk-11.0.12.7-hotspot\bin\java.exe' '@C:\Users\User\AppData\Local\Temp\cp_bmjpnsdon1bnc1o8uul0nco3p.argfile' 'lab.Main'
Database lab has been created successfully!
Database lab has been filled with customers successfully!
Database lab has been filled with workers successfully!
Database lab has been filled with cpu successfully!
Database lab has been filled with gpu successfully!
Database lab has been filled with orders successfully!
Printing customers...

ID : 2
First Name : Abel
Last Name : Wood
Email : Abel_Wood@gmail.com

ID : 4
First Name : Avery
Last Name : Schaefer
Email : Avery_Schaefer@gmail.com

ID : 1
First Name : Gale
Last Name : Bean
Email : Gale_Bean@gmail.com

ID : 3
First Name : Roosevelt
Last Name : Glover
Email : Roosevelt_Glover@gmail.com

ID : 5
First Name : Terry
Last Name : Duffy
Email : Terry_Duffy@gmail.com

Printing workers...

ID : 2
First Name : Abilene
Last Name : Debra
Position : senior
Email : Abilene_Debra@gmail.com

ID : 3
First Name : Caroline
Last Name : Jean
Position : junior
Email : Caroline_Jean@gmail.com
```

```
ID : 5
First Name : Joanie
Last Name : Julianne
Position : middle
Email : Joanie_Julianne@gmail.com

ID : 4
First Name : Kiki
Last Name : Gene
Position : middle
Email : Kiki_Gene@gmail.com

ID : 1
First Name : Lilac
Last Name : Rick
Position : junior
Email : Lilac_Rick@gmail.com

Printing CPU...

ID : 1
Name : AMD Ryzen 3 4100
Price : 99.0
Description : The AMD Ryzen 3 4100 is a desktop processor with 4 cores, launched in April 2022. It is part of
the Ryzen 3 lineup, using the Zen 2 (Renoir) architecture with Socket AM4.

ID : 3
Name : AMD FX-6300
Price : 132.0
Description : The AMD FX-6300 was a desktop processor with 6 cores, launched in October 2012. It is part of
the FX lineup, using the Vishera architecture with Socket AM3+. FX-6300 has 8MB of L3 cache and operates at
3.5 GHz by default, but can boost up to 4.1 GHz, depending on the workload.

ID : 4
Name : Intel Core i5-10400F
Price : 220.0
Description : The Intel Core i5-10400F is a desktop processor with 6 cores, launched in April 2020. It is part of
the Core i5 lineup, using the Comet Lake architecture with Socket 1200. Thanks to Intel Hyper-Threading the
core-count is effectively
doubled, to 12 threads. Core i5-10400F has 12MB of L3 cache and operates at 2.9 GHz by default, but can boost
up to 4.3 GHz.
ID : 2
Name : AMD Ryzen 7 5800X3D
Price : 449.0
Description : The AMD Ryzen 7 5800X3D is a desktop processor with 8 cores, launched in April 2022. It is part
of the Ryzen 7 lineup, using the Zen 3 (Vermeer) architecture with Socket AM4.

ID : 5
Name : Intel Core i9-12900K
Price : 599.0
Description : The Intel Core i9-12900K is a desktop processor with 16 cores, launched in November 2021. It is
part of the Core i9 lineup, using the Alder Lake-S architecture with Socket 1700. Thanks to Intel Hyper-
Threading the core-count is effectively doubled, to 24 threads. Core i9-12900K has 30MB of L3 cache and
operates at 3.2 GHz by default, but can boost up to 5.2 GHz.
```

```
Printing GPU...

ID : 3
Name : NVIDIA GeForce GTX 1060 6 GB
Price : 299.0
Description : The GeForce GTX 1060 6 GB was a performance-segment graphics card by NVIDIA, launched on July 19th, 2016. Built on the 16 nm process, and based on the GP106 graphics processor, in its GP106-400-A1 variant, the card supports DirectX 12.

ID : 1
Name : NVIDIA GeForce RTX 3060
Price : 329.0
Description : The GeForce RTX 3060 is a performance-segment graphics card by NVIDIA, launched on January 12th, 2021. Built on the 8 nm process, and based on the GA106 graphics processor, in its GA106-300-A1 variant, the card supports DirectX 12 Ultimate.

ID : 2
Name : AMD Radeon RX 6600 XT
Price : 379.0
Description : The Radeon RX 6600 XT is a performance-segment graphics card by AMD, launched on July 30th, 2021. Built on the 7 nm process, and based on the Navi 23 graphics processor, in its Navi 23 XT variant, the card supports DirectX 12 Ultimate.

ID : 4
Name : AMD Radeon RX 6700 XT
Price : 479.0
Description : The Radeon RX 6700 XT is a high-end graphics card by AMD, launched on March 3rd, 2021. Built on the 7 nm process, and based on the Navi 22 graphics processor, in its Navi 22 XT variant, the card supports DirectX 12 Ultimate.

ID : 5
Name : NVIDIA GeForce GTX 1080
Price : 599.0
Description : The GeForce GTX 1080 was a high-end graphics card by NVIDIA, launched on May 27th, 2016. Built on the 16 nm process, and based on the GP104 graphics processor, in its GP104-400-A1 variant, the card supports DirectX 12.

Printing orders...

ID : 1
Customer ID : 3
Worker ID : 2
Date : 2022-04-23
Price : 549.0
CPU ID : 4
GPU ID : 1

ID : 2
Customer ID : 5
Worker ID : 1
Date : 2022-04-23
Price : 748.0
CPU ID : 2
GPU ID : 3
```

```
ID : 3
Customer ID : 4
Worker ID : 5
Date : 2022-04-23
Price : 478.0
CPU ID : 1
GPU ID : 2

ID : 4
Customer ID : 2
Worker ID : 3
Date : 2022-04-23
Price : 1078.0
CPU ID : 5
GPU ID : 4

ID : 5
Customer ID : 1
Worker ID : 4
Date : 2022-04-23
Price : 731.0
CPU ID : 3
GPU ID : 5

Deleting last order...

Deleting last customer...

Deleting last worker...

Deleting last CPU...

Deleting last GPU...

Printing customers...

ID : 2
First Name : Abel
Last Name : Wood
Email : Abel_Wood@gmail.com

ID : 4
First Name : Avery
Last Name : Schaefer
Email : Avery_Schaefer@gmail.com

ID : 1
First Name : Gale
Last Name : Bean
Email : Gale_Bean@gmail.com

ID : 3
First Name : Roosevelt
Last Name : Glover
Email : Roosevelt_Glover@gmail.com
```

```
Printing workers...

ID : 2
First Name : Abilene
Last Name : Debra
Position : senior
Email : Abilene_Debra@gmail.com

ID : 3
First Name : Caroline
Last Name : Jean
Position : junior
Email : Caroline_Jean@gmail.com

ID : 4
First Name : Kiki
Last Name : Gene
Position : middle
Email : Kiki_Gene@gmail.com

ID : 1
First Name : Lilac
Last Name : Rick
Position : junior
Email : Lilac_Rick@gmail.com

Printing CPU...

ID : 1
Name : AMD Ryzen 3 4100
Price : 99.0
Description : The AMD Ryzen 3 4100 is a desktop processor with 4 cores, launched in April 2022. It is part of
the Ryzen 3 lineup, using the Zen 2 (Renoir) architecture with Socket AM4.

ID : 3
Name : AMD FX-6300
Price : 132.0
Description : The AMD FX-6300 was a desktop processor with 6 cores, launched in October 2012. It is part of
the FX lineup, using the Vishera architecture with Socket AM3+. FX-6300 has 8MB of L3 cache and operates at
3.5 GHz by default, but can boost up to 4.1 GHz, depending on the workload.

ID : 4
Name : Intel Core i5-10400F
Price : 220.0
Description : The Intel Core i5-10400F is a desktop processor with 6 cores, launched in April 2020. It is part of
the Core i5 lineup, using the Comet Lake architecture with Socket 1200. Thanks to Intel Hyper-Threading the
core-count is effectively
doubled, to 12 threads. Core i5-10400F has 12MB of L3 cache and operates at 2.9 GHz by default, but can boost
up to 4.3 GHz.
ID : 2
Name : AMD Ryzen 7 5800X3D
Price : 449.0
Description : The AMD Ryzen 7 5800X3D is a desktop processor with 8 cores, launched in April 2022. It is part
of the Ryzen 7 lineup, using the Zen 3 (Vermeer) architecture with Socket AM4.
```

```
Printing GPU...

ID : 3
Name : NVIDIA GeForce GTX 1060 6 GB
Price : 299.0
Description : The GeForce GTX 1060 6 GB was a performance-segment graphics card by NVIDIA, launched on
July 19th, 2016. Built on the 16 nm process, and based on the GP106 graphics processor, in its GP106-400-A1
variant, the card supports DirectX 12.

ID : 1
Name : NVIDIA GeForce RTX 3060
Price : 329.0
Description : The GeForce RTX 3060 is a performance-segment graphics card by NVIDIA, launched on January
12th, 2021. Built on the 8 nm process, and based on the GA106 graphics processor, in its GA106-300-A1 variant,
the card supports DirectX 12 Ultimate.

ID : 2
Name : AMD Radeon RX 6600 XT
Price : 379.0
Description : The Radeon RX 6600 XT is a performance-segment graphics card by AMD, launched on July 30th,
2021. Built on the 7 nm process, and based on the Navi 23 graphics processor, in its Navi 23 XT variant, the card
supports DirectX 12 Ultimate.

ID : 4
Name : AMD Radeon RX 6700 XT
Price : 479.0
Description : The Radeon RX 6700 XT is a high-end graphics card by AMD, launched on March 3rd, 2021. Built
on the 7 nm process, and based on the Navi 22 graphics processor, in its Navi 22 XT variant, the card supports
DirectX 12 Ultimate.

Printing orders...

ID : 1
Customer ID : 3
Worker ID : 2
Date : 2022-04-23
Price : 549.0
CPU ID : 4
GPU ID : 1

Increasing GPU prices...

Printing GPU...

ID : 3
Name : NVIDIA GeForce GTX 1060 6 GB
Price : 747.5
Description : The GeForce GTX 1060 6 GB was a performance-segment graphics card by NVIDIA, launched on
July 19th, 2016. Built on the 16 nm process, and based on the GP106 graphics processor, in its GP106-400-A1
variant, the card supports DirectX 12.
```

```
ID : 1
Name : NVIDIA GeForce RTX 3060
Price : 822.5
Description : The GeForce RTX 3060 is a performance-segment graphics card by NVIDIA, launched on January 12th, 2021. Built on the 8 nm process, and based on the GA106 graphics processor, in its GA106-300-A1 variant, the card supports DirectX 12 Ultimate.

ID : 2
Name : AMD Radeon RX 6600 XT
Price : 947.5
Description : The Radeon RX 6600 XT is a performance-segment graphics card by AMD, launched on July 30th, 2021. Built on the 7 nm process, and based on the Navi 23 graphics processor, in its Navi 23 XT variant, the card supports DirectX 12 Ultimate.

ID : 4
Name : AMD Radeon RX 6700 XT
Price : 1197.5
Description : The Radeon RX 6700 XT is a high-end graphics card by AMD, launched on March 3rd, 2021. Built on the 7 nm process, and based on the Navi 22 graphics processor, in its Navi 22 XT variant, the card supports DirectX 12 Ultimate.

PS D:\Documents\Visual Studio Code\Java\lab>
```

Вывод: Приобрел практические навыки разработки баз данных и начальной интеграции БД с кодом Java с помощью JDBC.