**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ЛАБОРАТОРНАЯ РАБОТА №2

по дисциплине: **«Визуальные средства разработки программных**

**приложений»**

на тему: **Наследование в *Java***

Выполнил: студент гр. ИТП-31

Дашкевич Д.А.

Принял: ассистент

Белявский Е. В.

Гомель 2019

**Цель**: изучить наследование в *Java*.

**Ход работы**

**Вариант 2**

# **Задание:**

С использованием механизма наследования в *Javа*, реализовать слой доступа к данным.

1. На основе сущностей предметной области создать классы их описывающие.

2. Информацию о предметной области хранить в БД (или любом другом хранилище: *XML*-файлы, текстовые, бинарные файлы, веб-сервисы и т.д. Тип хранилища выбирается студентом самостоятельно). Для доступа в случае с БД использовать *API* *JDBC* с использованием пула соединений, стандартного или разработанного самостоятельно. В качестве СУБД рекомендуется *MySQL* или *MSSqlserver*.

3. Доступ к данным реализовать с использованием шаблонов *FactoryMethod* или *AbstractFactory*.

2. Система Платежи. Клиент имеет одну или несколько Кредитных Карт, каждая из которых соответствует некоторому Счету в системе платежей. Клиент может при помощи Счета сделать Платеж, заблокировать Счет и пополнить Счет. Администратор снимает блокировку.

**Результат работы программы:**

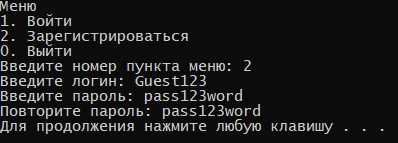
****

Рисунок 1 – Вывод в консоль процесса регистрации клиента

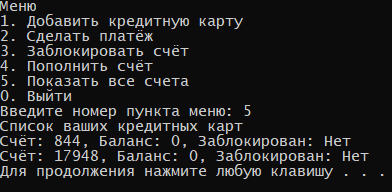


Рисунок 2 – Вывод в консоль результата всех счетов клиента

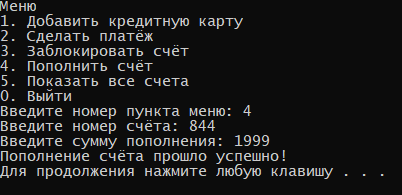


Рисунок 3 – Вывод в консоль результата пополнения счёта 844 на сумму 1999

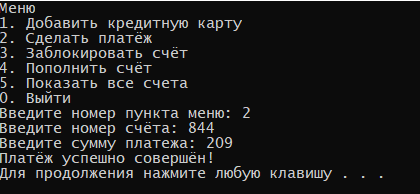


Рисунок 4 – Вывод в консоль результата совершения платежа на сумму 209 со счёта 844

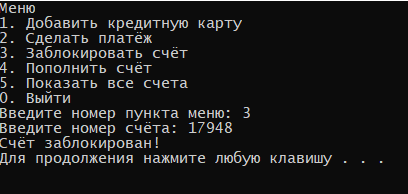


Рисунок 5 – Вывод в консоль результата блокировки счёта 17948

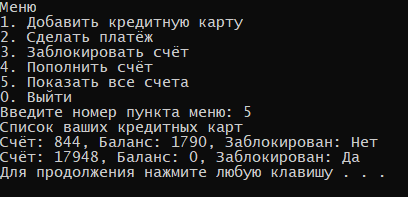


Рисунок 6 – Вывод в консоль значений счетов клиента после нескольких операций

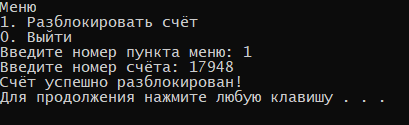


Рисунок 7 – Вывод в консоль результата разблокировки счёта 17948 администратором

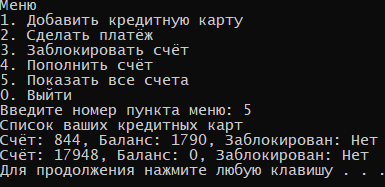


Рисунок 8 – Счета клиента после разблокировки счёта 17948

**Исходный код программы**

**Main.java:**

**public class** Main {  
 **public static void** main(String[] args) {  
 Menu menu = **new** Menu();  
 menu.openMenu();  
 }  
}

**Menu.java:**

**public class** Menu {  
 **private static final** String ***ADMIN\_LOGIN*** = **"admin"**;  
 **private static final** String ***ADMIN\_PASSWORD*** = **"admin"**;  
  
 **private static final** String ***ENTER\_LOGIN\_STR*** = **"Введите логин: "**;  
 **private static final** String ***ENTER\_PASSWORD\_STR*** = **"Введите пароль: "**;  
 **private static final** String ***ENTER\_ACCOUNT\_STR*** = **"Введите номер счёта: "**;  
  
 **private** Scanner **in** = **new** Scanner(System.***in***);  
  
 **private** PaymentSystem **paymentSystem**;  
 **private** Client **currentClient** = **null**;  
  
 **public** Menu() {  
 **paymentSystem** = PaymentSystemFactory.*getPaymentSystem*(PaymentSystemFactory.***MYSQL***);  
 }  
  
 **public void** openMenu() {  
 startMenuCycle();  
 }  
  
 **private long** inputLongValue() {  
 **long** k;  
 **try** {  
 k = **in**.nextLong();  
 } **catch** (InputMismatchException ex) {  
 ex.printStackTrace();  
 k = -1;  
 }  
 **return** k;  
 }  
  
 **private void** startMenuCycle() {  
 **while** (**true**) {  
 showStartMenu();  
 **int** k = (**int**) inputLongValue();  
 **switch** (k) {  
 **case** 0:  
 **paymentSystem**.saveChanges();  
 **return**;  
 **case** 1:  
 enter();  
 **break**;  
 **case** 2:  
 register();  
 **break**;  
 **default**:  
 System.***out***.println(**"Неправильный номер пункта меню!"**);  
 }  
 }  
 }  
  
 **private void** enter() {  
 System.***out***.print(***ENTER\_LOGIN\_STR***);  
 String login = **in**.next();  
 System.***out***.print(***ENTER\_PASSWORD\_STR***);  
 String password = **in**.next();  
 **if** (login.equals(***ADMIN\_LOGIN***) && password.equals(***ADMIN\_PASSWORD***)) {  
 adminMenuCycle();  
 **return**;  
 }  
 Client client = **paymentSystem**.findClientByLogin(login);  
 **if** (client == **null**) {  
 System.***out***.println(**"Клиента с таким логином не существует!"**);  
 } **else if** (client.getPassword().equals(password)) {  
 **currentClient** = client;  
 clientMenuCycle();  
 } **else** {  
 System.***out***.println(**"Неправильный пароль!"**);  
 }  
 }  
  
 **private void** register() {  
 String login, password, repeatedPassword;  
 **do** {  
 System.***out***.print(***ENTER\_LOGIN\_STR***);  
 login = **in**.next();  
 } **while** (**paymentSystem**.findClientByLogin(login) != **null**);  
 **do** {  
 System.***out***.print(***ENTER\_PASSWORD\_STR***);  
 password = **in**.next();  
 System.***out***.print(**"Повторите пароль: "**);  
 repeatedPassword = **in**.next();  
 } **while** (!password.equals(repeatedPassword));  
 **currentClient** = **new** Client(login, password);  
 **paymentSystem**.addClient(**currentClient**);  
 }  
  
 **private void** clientMenuCycle() {  
 **while** (**true**) {  
 showClientMenu();  
 **int** k = (**int**) inputLongValue();  
 **switch** (k) {  
 **case** 0:  
 **currentClient** = **null**;  
 **return**;  
 **case** 1:  
 **currentClient**.addCreditCard();  
 **break**;  
 **case** 2:  
 makePayment();  
 **break**;  
 **case** 3:  
 freezeAccount();  
 **break**;  
 **case** 4:  
 makeDeposit();  
 **break**;  
 **case** 5:  
 showAccounts();  
 **break**;  
 **default**:  
 System.***out***.println(**"Неправильный номер пункта меню!"**);  
 }  
 }  
 }  
  
 **private void** makePayment() {  
 System.***out***.print(***ENTER\_ACCOUNT\_STR***);  
 **long** account = inputLongValue();  
 CreditCard creditCard = **currentClient**.getCreditCardByAccount(account);  
 **if** (creditCard != **null**) {  
 System.***out***.print(**"Введите сумму платежа: "**);  
 **long** amount = inputLongValue();  
 **if** (creditCard.makePayment(amount)) {  
 System.***out***.println(**"Платёж успешно совершён!"**);  
 } **else** {  
 System.***out***.println(**"Платёж не совершён по неизвестной причине!"**);  
 }  
 } **else** {  
 System.***out***.println(**"Счёт не существует!"**);  
 }  
 }  
  
 **private void** freezeAccount() {  
 System.***out***.print(***ENTER\_ACCOUNT\_STR***);  
 **long** account = inputLongValue();  
 CreditCard creditCard = **currentClient**.getCreditCardByAccount(account);  
 **if** (creditCard != **null**) {  
 creditCard.freeze();  
 System.***out***.println(**"Счёт заблокирован!"**);  
 } **else** {  
 System.***out***.println(**"Счёт не существует!"**);  
 }  
 }  
  
 **private void** makeDeposit() {  
 System.***out***.print(***ENTER\_ACCOUNT\_STR***);  
 **long** account = inputLongValue();  
 CreditCard creditCard = **currentClient**.getCreditCardByAccount(account);  
 **if** (creditCard != **null**) {  
 System.***out***.print(**"Введите сумму пополнения: "**);  
 **long** amount = inputLongValue();  
 **if** (creditCard.makeDeposit(amount)) {  
 System.***out***.println(**"Пополнение счёта прошло успешно!"**);  
 } **else** {  
 System.***out***.println(**"Ошибка пополнения счёта!"**);  
 }  
 } **else** {  
 System.***out***.println(**"Счёт не существует!"**);  
 }  
 }  
  
 **private void** showAccounts() {  
 ArrayList<CreditCard> cards = **currentClient**.getCreditCards();  
 System.***out***.println(**"Список ваших кредитных карт"**);  
 **for** (CreditCard card : cards) {  
 System.***out***.printf(**"Счёт: %d, Баланс: %d, Заблокирован: %s\n"**, card.getAccount(),  
 card.getBalance(), card.isFreezed() ? **"Да"** : **"Нет"**);  
 }  
 }  
  
 **private void** adminMenuCycle() {  
 **while** (**true**) {  
 showAdminMenu();  
 **int** k = (**int**) inputLongValue();  
 **switch** (k) {  
 **case** 0:  
 **return**;  
 **case** 1:  
 unfreezeAccount();  
 **break**;  
 **default**:  
 System.***out***.println(**"Неправильный номер пункта меню!"**);  
 }  
 }  
 }  
  
 **private void** unfreezeAccount() {  
 System.***out***.print(***ENTER\_ACCOUNT\_STR***);  
 **long** account = inputLongValue();  
 CreditCard card = **paymentSystem**.findCreditCardByAccount(account);  
 **if** (card != **null**) {  
 card.unfreeze();  
 System.***out***.println(**"Счёт успешно разблокирован!"**);  
 } **else** {  
 System.***out***.println(**"Счёт не существует!"**);  
 }  
 }  
  
 **private void** show(MenuItems t) {  
 *pause*();  
 *clear*();  
 System.***out***.println(**"Меню"**);  
 t.showItems();  
 System.***out***.println(**"0. Выйти"**);  
 System.***out***.print(**"Введите номер пункта меню: "**);  
 }  
  
 **private void** showStartMenu() {  
 show(() -> {  
 System.***out***.println(**"1. Войти"**);  
 System.***out***.println(**"2. Зарегистрироваться"**);  
 });  
 }  
  
 **private void** showClientMenu() {  
 show(() -> {  
 System.***out***.println(**"1. Добавить кредитную карту"**);  
 System.***out***.println(**"2. Сделать платёж"**);  
 System.***out***.println(**"3. Заблокировать счёт"**);  
 System.***out***.println(**"4. Пополнить счёт"**);  
 System.***out***.println(**"5. Показать все счета"**);  
 });  
 }  
  
 **private void** showAdminMenu() {  
 show(() -> System.***out***.println(**"1. Разблокировать счёт"**));  
 }  
  
 **private static void** clear() {  
 **try** {  
 **new** ProcessBuilder(**"cmd"**, **"/c"**, **"cls"**).inheritIO().start().waitFor();  
 } **catch** (IOException | InterruptedException ex) {  
 System.***out***.println(ex.getMessage());  
 }  
 }  
  
 **private static void** pause() {  
 **try** {  
 **new** ProcessBuilder(**"cmd"**, **"/c"**, **"pause"**).inheritIO().start().waitFor();  
 } **catch** (IOException | InterruptedException ex) {  
 System.***out***.println(ex.getMessage());  
 }  
 }  
  
  
 **private interface** MenuItems {  
 **void** showItems();  
 }  
}

**CreditCard.java:**

**public class** CreditCard **implements** Serializable {  
 **private int id** = -1;  
 **private long account**;  
 **private long balance** = 0;  
 **private boolean freezed** = **false**;  
  
 CreditCard() {  
 **this**.**account** = Math.*abs*(**new** Random().nextInt(100000));  
 }  
  
 **private** CreditCard(**long** account, **long** balance, **boolean** freezed) {  
 **this**.**account** = account;  
 **this**.**balance** = balance;  
 **this**.**freezed** = freezed;  
 }  
  
 **public** CreditCard(**int** id, **long** account, **long** balance, **boolean** freezed) {  
 **this**(account, balance, freezed);  
 **this**.**id** = id;  
 }  
  
 **public boolean** makePayment(**long** amount) {  
 **if** (**balance** - amount < 0 || amount <= 0 || **freezed**) {  
 **return false**;  
 }  
 **balance** -= amount;  
 **return true**;  
 }  
  
 **public void** freeze() {  
 **freezed** = **true**;  
 }  
  
 **public void** unfreeze() {  
 **freezed** = **false**;  
 }  
  
 **public void** setId(**int** id) {  
 **this**.**id** = id;  
 }  
  
 **public int** getId() {  
 **return id**;  
 }  
  
 **public long** getAccount() {  
 **return account**;  
 }  
  
 **public boolean** makeDeposit(**long** amount) {  
 **if** (amount <= 0) {  
 **return false**;  
 }  
 **balance** += amount;  
 **return true**;  
 }  
  
 **public long** getBalance() {  
 **return balance**;  
 }  
  
 **public boolean** isFreezed() {  
 **return freezed**;  
 }  
}

**Client.java:**

**public class** Client **implements** Serializable {  
 **private int id** = -1;  
 **private** String **login**;  
 **private** String **password**;  
 **private** ArrayList<CreditCard> **creditCards**;  
  
 **public** Client(String login, String password) {  
 **this**.**login** = login;  
 **this**.**password** = password;  
 **this**.**creditCards** = **new** ArrayList<>();  
 }  
  
 **public** Client(**int** id, String login, String password) {  
 **this**(login, password);  
 **this**.**id** = id;  
 }  
  
 **public void** addCreditCard() {  
 **creditCards**.add(**new** CreditCard());  
 }  
  
 **public void** addCreditCard(CreditCard creditCard) {  
 **creditCards**.add(creditCard);  
 }  
  
 **public** CreditCard getCreditCardByAccount(**long** account) {  
 **for** (CreditCard creditCard : **creditCards**) {  
 **if** (creditCard.getAccount() == account) {  
 **return** creditCard;  
 }  
 }  
 **return null**;  
 }  
  
 **public void** setId(**int** id) {  
 **this**.**id** = id;  
 }  
  
 **public int** getId() {  
 **return id**;  
 }  
  
 **public** String getLogin() {  
 **return login**;  
 }  
  
 **public** String getPassword() {  
 **return password**;  
 }  
  
 **public** ArrayList<CreditCard> getCreditCards() {  
 **return creditCards**;  
 }  
}

**PaymentSystem.java:**

**public interface** PaymentSystem {  
 Client findClientByLogin(String login);  
  
 CreditCard findCreditCardByAccount(**long** account);  
  
 **void** addClient(Client client);  
  
 **void** saveChanges();  
}

**AbstractPaymentSystem.java:**

**public abstract class** AbstractPaymentSystem **implements** PaymentSystem {  
 **protected** ArrayList<Client> **clients**;  
  
 **protected abstract void** loadClients();  
  
 **protected abstract void** saveClients();  
  
 **public** AbstractPaymentSystem() {  
 **this**.**clients** = **new** ArrayList<>();  
 }  
  
 @Override  
 **public void** saveChanges() {  
 saveClients();  
 }  
  
 @Override  
 **public** Client findClientByLogin(String login) {  
 **for** (Client client : **clients**) {  
 **if** (client.getLogin().equals(login)) {  
 **return** client;  
 }  
 }  
 **return null**;  
 }  
  
 @Override  
 **public** CreditCard findCreditCardByAccount(**long** account) {  
 **for** (Client client : **clients**) {  
 CreditCard card = client.getCreditCardByAccount(account);  
 **if** (card != **null**) {  
 **return** card;  
 }  
 }  
 **return null**;  
 }  
}

**SimpleFilePaymentSystem.java:**

**public class** SimpleFilePaymentSystem **extends** AbstractPaymentSystem {  
  
 **private** String **filename**;  
  
 SimpleFilePaymentSystem() {  
 **super**();  
 **filename** = PropertiesSystem.*SIMPLE\_FILENAME*;  
 loadClients();  
 }  
  
 @Override  
 **public void** addClient(Client client) {  
 **clients**.add(client);  
 }  
  
 @Override  
 **protected void** loadClients() {  
 File file = **new** File(**filename**);  
 **if** (file.exists()) {  
 **try** {  
 ObjectInputStream stream = **new** ObjectInputStream(**new** FileInputStream(file));  
 **int** size = stream.readInt();  
 **clients** = **new** ArrayList<>(size);  
 **for** (**int** i = 0; i < size; i++) {  
 **clients**.add((Client) stream.readObject());  
 }  
 stream.close();  
 } **catch** (IOException | ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
 @Override  
 **protected void** saveClients() {  
 File file = **new** File(**filename**);  
 **if** (!file.exists()) {  
 **try** {  
 **if** (!file.createNewFile()) {  
 **throw new** IOException(**"Ошибка при создании файла!"**);  
 }  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 **try** {  
 ObjectOutputStream stream = **new** ObjectOutputStream(**new** FileOutputStream(file));  
 stream.writeInt(**clients**.size());  
 **for** (Client client : **clients**) {  
 stream.writeObject(client);  
 }  
 stream.close();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}

**MySqlPaymentSystem.java:**

**public class** MySqlPaymentSystem **extends** AbstractPaymentSystem {  
  
 **private** String **url**;  
 **private** String **username**;  
 **private** String **password**;  
  
 **private** Connection **connection** = **null**;  
  
 MySqlPaymentSystem() {  
 **super**();  
 **this**.**url** = PropertiesSystem.*MYSQL\_CONNECTION*;  
 **this**.**username** = PropertiesSystem.*MYSQL\_USERNAME*;  
 **this**.**password** = PropertiesSystem.*MYSQL\_PASSWORD*;  
 loadClients();  
 }  
  
 **private** Connection getConnection() **throws** SQLException {  
 **if** (**connection** == **null**) {  
 **connection** = DriverManager.*getConnection*(**url**, **username**, **password**);  
 }  
 **return connection**;  
 }  
  
 **private void** closeConnection() {  
 **if** (**connection** != **null**) {  
 **try** {  
 **connection**.close();  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
 **connection** = **null**;  
 }  
  
 @Override  
 **public void** saveChanges() {  
 **super**.saveChanges();  
 closeConnection();  
 }  
  
 @Override  
 **protected void** loadClients() {  
 **try** (Statement statement = getConnection().createStatement()) {  
 **try** (ResultSet resultSet = statement.executeQuery(PropertiesSystem.*SELECT\_CLIENTS\_SQL*)) {  
 **while** (resultSet.next()) {  
 **int** id = resultSet.getInt(**"Id"**);  
 Client client = **new** Client(id, resultSet.getString(**"Login"**),  
 resultSet.getString(**"Password"**));  
 **clients**.add(client);  
 }  
 }  
 } **catch** (SQLException ex) {  
 ex.printStackTrace();  
 }  
 loadCreditCards();  
 }  
  
 **private void** loadCreditCards() {  
 **try** (PreparedStatement statement = getConnection().prepareStatement(PropertiesSystem.*SELECT\_CREDITCARDS\_SQL*)) {  
 **try** (ResultSet resultSet = statement.executeQuery()) {  
 **while** (resultSet.next()) {  
 **int** clientId = resultSet.getInt(**"ClientsId"**);  
 **for** (Client client : **clients**) {  
 **if** (client.getId() == clientId) {  
 client.addCreditCard(**new** CreditCard(resultSet.getInt(**"Id"**),  
 resultSet.getLong(**"Account"**), resultSet.getLong(**"Balance"**),  
 resultSet.getBoolean(**"Freezed"**)));  
 **break**;  
 }  
 }  
 }  
 }  
 } **catch** (SQLException ex) {  
 ex.printStackTrace();  
 }  
 }  
  
 @Override  
 **public void** addClient(Client client) {  
 **if** (insertClient(client) != -1) {  
 **clients**.add(client);  
 }  
 }  
  
 **private int** insertClient(Client client) {  
 **int** returnId = -1;  
 **try** (PreparedStatement statement = getConnection()  
 .prepareStatement(PropertiesSystem.*INSERT\_CLIENT\_SQL*, **new** String[]{**"ID"**})) {  
 statement.setString(1, client.getLogin());  
 statement.setString(2, client.getPassword());  
 statement.executeUpdate();  
 **try** (ResultSet resultSet = statement.getGeneratedKeys()) {  
 **int** insertId = 0;  
 **while** (resultSet.next()) {  
 insertId = resultSet.getInt(1);  
 }  
 returnId = insertId;  
 client.setId(insertId);  
 }  
 } **catch** (SQLException ex) {  
 ex.printStackTrace();  
 }  
 **return** returnId;  
 }  
  
 @Override  
 **protected void** saveClients() {  
 updateClients();  
 updateCreditCards();  
 }  
  
 **private void** updateClients() {  
 **for** (Client client : **clients**) {  
 updateClient(client);  
 }  
 }  
  
 **private void** updateCreditCards() {  
 **for** (Client client : **clients**) {  
 **for** (CreditCard creditCard : client.getCreditCards()) {  
 **if** (creditCard.getId() == -1) {  
 insertCreditCard(creditCard, client.getId());  
 } **else** {  
 updateCreditCard(creditCard);  
 }  
 }  
 }  
 }  
  
 **private void** updateClient(Client client) {  
 **try** (PreparedStatement statement = getConnection().prepareStatement(PropertiesSystem.*UPDATE\_CLIENT\_SQL*)) {  
 statement.setString(1, client.getLogin());  
 statement.setString(2, client.getPassword());  
 statement.setInt(3, client.getId());  
 statement.executeUpdate();  
 } **catch** (SQLException ex) {  
 ex.printStackTrace();  
 }  
 }  
  
 **private void** insertCreditCard(CreditCard creditCard, **int** id) {  
 **try** (PreparedStatement statement = getConnection().prepareStatement(PropertiesSystem.*INSERT\_CREDITCARD\_SQL*, **new** String[]{**"ID"**})) {  
 statement.setLong(1, creditCard.getAccount());  
 statement.setLong(2, creditCard.getBalance());  
 statement.setBoolean(3, creditCard.isFreezed());  
 statement.setInt(4, id);  
 statement.executeUpdate();  
 **try** (ResultSet resultSet = statement.getGeneratedKeys()) {  
 **int** insertId = 0;  
 **while** (resultSet.next()) {  
 insertId = resultSet.getInt(1);  
 }  
 creditCard.setId(insertId);  
 }  
 } **catch** (SQLException ex) {  
 ex.printStackTrace();  
 }  
 }  
  
 **private void** updateCreditCard(CreditCard creditCard) {  
 **try** (PreparedStatement statement = getConnection().prepareStatement(PropertiesSystem.*UPDATE\_CREDITCARD\_SQL*)) {  
 statement.setLong(1, creditCard.getAccount());  
 statement.setLong(2, creditCard.getBalance());  
 statement.setBoolean(3, creditCard.isFreezed());  
 statement.setInt(4, creditCard.getId());  
 statement.executeUpdate();  
 } **catch** (SQLException ex) {  
 ex.printStackTrace();  
 }  
 }  
}

**PaymentSystemFactory.java:**

**public class** PaymentSystemFactory {  
  
 **public static final int *SIMPLEFILE*** = 1;  
 **public static final int *MYSQL*** = 2;  
  
 **public static** PaymentSystem getPaymentSystem(**int** code) {  
 **switch** (code) {  
 **case *SIMPLEFILE***:  
 **return new** SimpleFilePaymentSystem();  
 **case *MYSQL***:  
 **return new** MySqlPaymentSystem();  
 **default**:  
 **return null**;  
 }  
 }  
}

**PropertiesSystem.java:**

**public class** PropertiesSystem {  
  
 **private static final** String ***DB\_RESOURCES\_FILEPATH*** = **"src/util/db\_resources.properties"**;  
  
 **public static** String *SIMPLE\_FILENAME* = **"client\_database.txt"**;  
  
 **public static** String *MYSQL\_CONNECTION* = **"jdbc:mysql://localhost:3306/javadb?useLegacyDatetimeCode=false&serverTimezone=UTC"**;  
 **public static** String *MYSQL\_USERNAME* = **"root"**;  
 **public static** String *MYSQL\_PASSWORD* = **"root"**;  
  
 **public static** String *SELECT\_CLIENTS\_SQL* = **"SELECT \* FROM clients;"**;  
 **public static** String *SELECT\_CREDITCARDS\_SQL* = **"SELECT \* FROM creditcards;"**;  
 **public static** String *INSERT\_CLIENT\_SQL* = **"INSERT INTO clients(Login, Password) VALUES (?, ?);"**;  
 **public static** String *INSERT\_CREDITCARD\_SQL* = **"INSERT INTO creditcards(Account, Balance, Freezed, ClientsId) VALUES (?, ?, ?, ?);"**;  
 **public static** String *UPDATE\_CLIENT\_SQL* = **"UPDATE clients SET Login = ?, Password = ? WHERE Id = ?;"**;  
 **public static** String *UPDATE\_CREDITCARD\_SQL* = **"UPDATE creditcards SET Account = ?, Balance = ?, Freezed = ? WHERE Id = ?;"**;  
  
  
 **static** {  
 **try** (FileInputStream inputStream = **new** FileInputStream(***DB\_RESOURCES\_FILEPATH***)) {  
 Properties properties = **new** Properties();  
 properties.load(inputStream);  
  
 *SIMPLE\_FILENAME* = properties.getProperty(**"SIMPLE\_FILENAME"**);  
  
 *MYSQL\_CONNECTION* = properties.getProperty(**"MYSQL\_CONNECTION"**);  
 *MYSQL\_USERNAME* = properties.getProperty(**"MYSQL\_USERNAME"**);  
 *MYSQL\_PASSWORD* = properties.getProperty(**"MYSQL\_PASSWORD"**);  
  
 *SELECT\_CLIENTS\_SQL* = properties.getProperty(**"SELECT\_CLIENTS\_SQL"**);  
 *SELECT\_CREDITCARDS\_SQL* = properties.getProperty(**"SELECT\_CREDITCARDS\_SQL"**);  
 *INSERT\_CLIENT\_SQL* = properties.getProperty(**"INSERT\_CLIENT\_SQL"**);  
 *INSERT\_CREDITCARD\_SQL* = properties.getProperty(**"INSERT\_CREDITCARD\_SQL"**);  
 *UPDATE\_CLIENT\_SQL* = properties.getProperty(**"UPDATE\_CLIENT\_SQL"**);  
 *UPDATE\_CREDITCARD\_SQL* = properties.getProperty(**"UPDATE\_CREDITCARD\_SQL"**);  
 } **catch** (IOException ex) {  
 ex.printStackTrace();  
 }  
 }  
}

**Вывод:** в результате выполнения лабораторной работы было изучено наследование в *Java*, разработана программа, которая работает с базами данных.