МІНІСТЕРСТВО ОСВІТИ І НАУКИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра комп’ютеризованих систем автоматики



**Лабораторна робота №7**

з курсу:

**“Організація баз даних та знань”**

Виконала:  
ст.гр.  ІР-23

Криворучка Ольга-Анна

Перевірила:

Асист. каф.

Лагун І. І.

Львів – 2018

**ЗАВДАННЯ**

1. Розробити базу даних (БД). БД може бути на довільну тему, наприклад, домашня бібліотека, фонотека, колекції марок, листівок, мої друзі і т.д. При цьому БД має бути унікальною і не повторювати БД інших студентів (при плагіаті робота не зараховується обом).
2. БД повинна розгортатися за допомогою SQL-скріпта.
3. Заповнити кожну таблицю БД як мінімум по 10 записів (якщо кількість звісно не обмежується логікою).
4. Реалізувати збережувані процедури для вставки даних у таблиці БД, що містять відповідну логіку щодо цілісності та коректності даних.
5. Клієнтська програма мовою Java створюється у вигляді Maven проекту з підключенням до MySQL.
6. Робота у програмі реалізовується з використанням меню у консолі.
7. Програма повинна забезпечувати роботу з БД за допомогою Hibernate:

* вивід даних з таблиць;
* вставку даних у таблиці (через INSERT);
* видалення даних з таблиці;
* обновлення даних у таблицях.
* обов’язково реалізувати вивід даних зі стикувальної таблиці зв’язку М:М, тобто вивести для кожного суб’єкта з одної таблиці усі суб’єкти другої таблиці, які приєднані до нього.
* реалізувати вставку/видалення в/зі стикувальної таблиці зв’язку М:М.

**CustomerEntity.java**

package com.kryvoruchka;  
  
import javax.persistence.\*;  
import java.util.List;  
  
@Entity  
@Table(name = "customer", schema = "onlineshop")  
public class CustomerEntity {  
 private String firstName;  
 private String lastName;  
 private String adress;  
 private int passName;  
 private String payment;  
 private List<ClothesEntity> clothes;  
  
 public CustomerEntity() {  
 }  
  
 public CustomerEntity(String firstName, String lastName, String adress, int passName, String payment) {  
 this.firstName = firstName;  
 this.lastName = lastName;  
 this.adress = adress;  
 this.passName = passName;  
 this.payment = payment;  
 }  
  
 @Basic  
 @Column(name = "first\_name")  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public void setFirstName(String firstName) {  
 this.firstName = firstName;  
 }  
  
 @Basic  
 @Column(name = "last\_name")  
 public String getLastName() {  
 return lastName;  
 }  
  
 public void setLastName(String lastName) {  
 this.lastName = lastName;  
 }  
  
 @Basic  
 @Column(name = "adress")  
 public String getAdress() {  
 return adress;  
 }  
  
 public void setAdress(String adress) {  
 this.adress = adress;  
 }  
  
 @Id  
 @Column(name = "pass\_name")  
 public int getPassName() {  
 return passName;  
 }  
  
 public void setPassName(int passName) {  
 this.passName = passName;  
 }  
  
 @Basic  
 @Column(name = "payment")  
 public String getPayment() {  
 return payment;  
 }  
  
 public void setPayment(String payment) {  
 this.payment = payment;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
  
 CustomerEntity that = (CustomerEntity) o;  
  
 if (passName != that.passName) return false;  
 if (firstName != null ? !firstName.equals(that.firstName) : that.firstName != null) return false;  
 if (lastName != null ? !lastName.equals(that.lastName) : that.lastName != null) return false;  
 if (adress != null ? !adress.equals(that.adress) : that.adress != null) return false;  
 if (payment != null ? !payment.equals(that.payment) : that.payment != null) return false;  
  
 return true;  
 }  
  
 @Override  
 public int hashCode() {  
 int result = firstName != null ? firstName.hashCode() : 0;  
 result = 31 \* result + (lastName != null ? lastName.hashCode() : 0);  
 result = 31 \* result + (adress != null ? adress.hashCode() : 0);  
 result = 31 \* result + passName;  
 result = 31 \* result + (payment != null ? payment.hashCode() : 0);  
 return result;  
 }  
  
 @ManyToMany(mappedBy = "customers")  
 public List<ClothesEntity> getClothes() {  
 return clothes;  
 }  
  
 public void addClothesEntity(ClothesEntity clothesEntity) {  
 if (!getClothes().contains(clothesEntity)) {  
 getClothes().add(clothesEntity);  
 }  
 if (!clothesEntity.getCustomers().contains(this)) {  
 clothesEntity.getCustomers().add(this);  
 }  
 }  
  
 public void setClothes(List<ClothesEntity> clothes) {  
 this.clothes = clothes;  
 }  
}

**MakerEntity.java**

package com.kryvoruchka;  
  
import javax.persistence.\*;  
import java.util.Collection;  
import java.util.List;  
  
@Entity  
@Table(name = "maker", schema = "onlineshop", catalog = "")  
public class MakerEntity {  
 private String companyName;  
 private String firstName;  
 private String lastName;  
 private int price;  
 private List<ClothesEntity> clothesByMaker;  
  
 public MakerEntity() {  
 }  
  
 public MakerEntity(String companyName, String firstName, String lastName, int price) {  
 this.companyName = companyName;  
 this.firstName = firstName;  
 this.lastName = lastName;  
 this.price = price;  
 }  
  
 public MakerEntity(String companyName) {  
 this.companyName = companyName;  
 }  
  
 @Id  
 @Column(name = "company\_name")  
 public String getCompanyName() {  
 return companyName;  
 }  
  
 public void setCompanyName(String companyName) {  
 this.companyName = companyName;  
 }  
  
 @Basic  
 @Column(name = "first\_name")  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public void setFirstName(String firstName) {  
 this.firstName = firstName;  
 }  
  
 @Basic  
 @Column(name = "last\_name")  
 public String getLastName() {  
 return lastName;  
 }  
  
 public void setLastName(String lastName) {  
 this.lastName = lastName;  
 }  
  
 @Basic  
 @Column(name = "price")  
 public int getPrice() {  
 return price;  
 }  
  
 public void setPrice(int price) {  
 this.price = price;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
  
 MakerEntity that = (MakerEntity) o;  
  
 if (price != that.price) return false;  
 if (companyName != null ? !companyName.equals(that.companyName) : that.companyName != null) return false;  
 if (firstName != null ? !firstName.equals(that.firstName) : that.firstName != null) return false;  
 if (lastName != null ? !lastName.equals(that.lastName) : that.lastName != null) return false;  
  
 return true;  
 }  
  
 @Override  
 public int hashCode() {  
 int result = companyName != null ? companyName.hashCode() : 0;  
 result = 31 \* result + (firstName != null ? firstName.hashCode() : 0);  
 result = 31 \* result + (lastName != null ? lastName.hashCode() : 0);  
 result = 31 \* result + price;  
 return result;  
 }  
  
 @OneToMany(mappedBy = "fkClothesMaker")  
 public List<ClothesEntity> getClothesByMaker() {  
 return clothesByMaker;  
 }  
  
 public void setClothesByMaker(List<ClothesEntity> clothesByMaker) {  
 this.clothesByMaker = clothesByMaker;  
 }  
}

**ClothesEntity.java**

package com.kryvoruchka;  
  
import javax.persistence.\*;  
import java.util.List;  
  
@Entity  
@Table(name = "clothes", schema = "onlineshop")  
public class ClothesEntity {  
 private int id;  
 private String type;  
 private int amount;  
 private String color;  
 private String size;  
 private MakerEntity fkClothesMaker;  
 private List<CustomerEntity> customers;  
  
 public ClothesEntity() {  
 }  
  
 public ClothesEntity(String type, int amount, String color, String size, MakerEntity fkClothesMaker) {  
 this.type = type;  
 this.amount = amount;  
 this.color = color;  
 this.size = size;  
 this.fkClothesMaker = fkClothesMaker;  
 }  
  
 @Id  
 @Column(name = "id")  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 @Basic  
 @Column(name = "type")  
 public String getType() {  
 return type;  
 }  
  
 public void setType(String type) {  
 this.type = type;  
 }  
  
 @Basic  
 @Column(name = "amount")  
 public int getAmount() {  
 return amount;  
 }  
  
 public void setAmount(int amount) {  
 this.amount = amount;  
 }  
  
 @Basic  
 @Column(name = "color")  
 public String getColor() {  
 return color;  
 }  
  
 public void setColor(String color) {  
 this.color = color;  
 }  
  
 @Basic  
 @Column(name = "size")  
 public String getSize() {  
 return size;  
 }  
  
 public void setSize(String size) {  
 this.size = size;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
  
 ClothesEntity that = (ClothesEntity) o;  
  
 if (id != that.id) return false;  
 if (amount != that.amount) return false;  
 if (type != null ? !type.equals(that.type) : that.type != null) return false;  
 if (color != null ? !color.equals(that.color) : that.color != null) return false;  
 if (size != null ? !size.equals(that.size) : that.size != null) return false;  
  
 return true;  
 }  
  
 @Override  
 public int hashCode() {  
 int result = id;  
 result = 31 \* result + (type != null ? type.hashCode() : 0);  
 result = 31 \* result + amount;  
 result = 31 \* result + (color != null ? color.hashCode() : 0);  
 result = 31 \* result + (size != null ? size.hashCode() : 0);  
 return result;  
 }  
  
 @ManyToOne  
 @JoinColumn(name = "FK\_clothes\_maker", referencedColumnName = "company\_name", nullable = false)  
 public MakerEntity getFkClothesMaker() {  
 return fkClothesMaker;  
 }  
  
 public void setFkClothesMaker(MakerEntity fkClothesMaker) {  
 this.fkClothesMaker = fkClothesMaker;  
 }  
  
 @ManyToMany  
 @JoinTable(name = "clothes\_has\_customer", schema = "onlineshop", joinColumns = @JoinColumn(name = "clothes\_id",  
 referencedColumnName = "id", nullable = false), inverseJoinColumns = @JoinColumn(name =  
 "customer\_pass\_name", referencedColumnName = "pass\_name", nullable = false))  
 public List<CustomerEntity> getCustomers() {  
 return customers;  
 }  
  
 public void setCustomers(List<CustomerEntity> customers) {  
 this.customers = customers;  
 }  
}

**View.java**

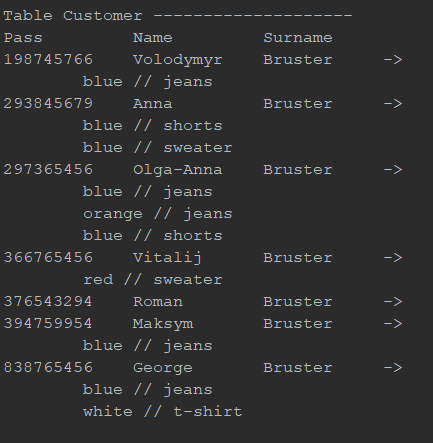
package com.kryvoruchka;  
  
import org.hibernate.HibernateException;  
import org.hibernate.Session;  
import org.hibernate.SessionFactory;  
import org.hibernate.cfg.Configuration;  
  
import java.util.\*;  
  
public class View {  
 private Map<String, String> menu;  
 private static Scanner *input* = new Scanner(System.*in*, "UTF-8");  
 private static final SessionFactory *sessionFactory*;  
  
 public View() {  
 menu = new LinkedHashMap<>();  
 menu.put("1", " 1 - Read Data From A Table");  
 menu.put("2", " 2 - Read Join Data From CustomerClothes");  
 menu.put("3", " 3 - Update Customer Name");  
 menu.put("4", " 4 - Insert To Table");  
 menu.put("5", " 5 - Insert Clothes For Customer");  
 menu.put("6", " 5 - Delete Customer By Name");  
 menu.put("E", " E - exit");  
 }  
  
 private void outputMenu() {  
 System.*out*.print("\nMENU:\n");  
 for (String str : menu.values()) {  
 System.*out*.print(str + "\n");  
 }  
 }  
  
 static {  
 try {  
 *sessionFactory* = new Configuration().configure().buildSessionFactory();  
 } catch (Throwable ex) {  
 throw new ExceptionInInitializerError(ex);  
 }  
 }  
  
 public static Session getSession() throws HibernateException {  
 return *sessionFactory*.openSession();  
 }  
  
 private void manager(final String num) {  
 try (Session session = *getSession*()) {  
  
  
 switch (num) {  
 case "1": {  
 System.*out*.println("Input a table name: ");  
 String table = *input*.nextLine();  
 EntityToDB.*ReadAllTable*(session, table);  
 break;  
 }  
 case "2": {  
 EntityToDB.*ReadClothesOfCustomer*(session);  
 break;  
 }  
 case "3": {  
 EntityToDB.*updateCustomerName*(session);  
 break;  
 }  
 case "4": {  
 System.*out*.println("Input a table name: ");  
 String table = *input*.nextLine();  
 if (table.equalsIgnoreCase("clothes")) {  
 EntityToDB.*insertClothes*(session);  
 } else if (table.equalsIgnoreCase("maker")) {  
 EntityToDB.*insertMaker*(session);  
 } else if (table.equalsIgnoreCase("customer")){  
 EntityToDB.*insertCustomer*(session);  
 }  
 else {  
 System.*out*.println("Error, there is no table with this name");  
 }  
 break;  
 }  
 case "5": {  
 EntityToDB.*AddClothesForCustomer*(session);  
 break;  
 }  
 case "6": {  
 EntityToDB.*deleteCustomerByName*(session);  
 break;  
 }  
 case "E": {  
 System.*out*.println(" Goodbye!!!");  
 return;  
 }  
 default: {  
 System.*out*.println("Error! Menu has not this point");  
 }  
 }  
 } catch (ClassNotFoundException e) {  
 System.*out*.println("Invalid class name");  
 }  
 }  
  
 public final void show() {  
 String keyMenu;  
 do {  
 outputMenu();  
 System.*out*.println("Please, select menu point");  
 keyMenu = *input*.nextLine().toUpperCase();  
 manager(keyMenu);  
 do {  
 System.*out*.println("\n M - return menu\n E - exit");  
 keyMenu = *input*.nextLine().toUpperCase();  
 if (keyMenu.equalsIgnoreCase("E")) {  
 manager(keyMenu);  
 return;  
 }  
 } while (!keyMenu.equalsIgnoreCase("M"));  
  
 } while (!keyMenu.equalsIgnoreCase("E"));  
 }  
}

**EntityToDB.java**

package com.kryvoruchka;  
  
import org.hibernate.query.Query;  
import org.hibernate.Session;  
import org.hibernate.SessionFactory;  
import org.hibernate.cfg.Configuration;  
  
import java.util.\*;  
  
public class EntityToDB {  
 public static void ReadAllTable(Session session, String table) throws ClassNotFoundException {  
 table = table.substring(0, 1).toUpperCase() + table.substring(1);  
  
 Query query = session.createQuery(new StringBuilder().append("from ").append(table).append("Entity").toString());  
 System.*out*.format("\nTable " + table + " --------------------\n");  
 SessionFactory sessionFactory = new Configuration().configure().buildSessionFactory();  
 List<String> columnNames = new ArrayList<>(Arrays.*asList*(sessionFactory.getClassMetadata(Class.*forName*("com.kryvoruchka" +  
 "." + table + "Entity")).getPropertyNames()));  
 String str = sessionFactory.getClassMetadata(Class.*forName*("com.kryvoruchka." + table + "Entity"))  
 .getIdentifierPropertyName();  
 columnNames.add(str);  
  
 for (String s : columnNames) {  
 System.*out*.format("%-15s", s);  
 }  
 System.*out*.println();  
  
 for (Object obj : query.list()) {  
 if (table.equalsIgnoreCase("maker")) {  
 MakerEntity c = (MakerEntity) obj;  
 if (c.getClothesByMaker().size() != 0) {  
 System.*out*.format("%-14s %-14s %-14s %-14s %-14s\n", c.getClothesByMaker().iterator().next().getType(), c  
 .getFirstName(), c.getLastName(), c.getPrice(), c.getCompanyName());  
 } else {  
 System.*out*.format("%-14s %-14s %-14s %-14s %-14s\n", "---", c.getFirstName(), c.getLastName(), c  
 .getPrice(), c.getCompanyName());  
 }  
  
 } else if (table.equalsIgnoreCase("customer")) {  
 CustomerEntity c = (CustomerEntity) obj;  
 if (c.getClothes().size() != 0) {  
 System.*out*.format("%-14s %-14s %-14s %-14s %-14s %-14s\n", c.getAdress(), c.getClothes().iterator()  
 .next().getType(), c.getFirstName(), c.getLastName(), c.getPayment(), c  
 .getPassName());  
 } else {  
 System.*out*.format("%-14s %-14s %-14s %-14s %-14s %-14s\n", c.getAdress(), "---", c.getFirstName(), c  
 .getLastName(), c.getPayment(), c.getPassName());  
 }  
  
 } else if (table.equalsIgnoreCase("clothes")) {  
 ClothesEntity c = (ClothesEntity) obj;  
 if (c.getCustomers().size() != 0) {  
 System.*out*.format("%-14s %-14s %-14s %-14s %-14s %-14s %-45s\n", c.getAmount(), c.getColor(), c  
 .getCustomers().iterator().next().getFirstName(), c.getFkClothesMaker().getFirstName(), c.getSize(), c  
 .getType(), c.getId());  
 } else {  
 System.*out*.format("%-14s %-14s %-14s %-14s %-14s %-14s %-45s\n", c.getAmount(), c.getColor(),  
 "---", c.getFkClothesMaker().getFirstName(), c.getSize(), c.getType(), c.getId());  
 }  
  
 }  
 }  
 }  
  
 public static void ReadClothesOfCustomer(Session session) {  
 Query query = session.createQuery("from " + "CustomerEntity");  
 System.*out*.format("\nTable Customer --------------------\n");  
 System.*out*.format("%-12s %-12s %-12s \n", "Pass", "Name", "Surname");  
 for (Object obj : query.list()) {  
 CustomerEntity customer = (CustomerEntity) obj;  
 System.*out*.format("%-12d %-12s %-12s->\n", customer.getPassName(), customer.getFirstName(), customer  
 .getLastName());  
 for (ClothesEntity clothes : customer.getClothes()) {  
 System.*out*.format("\t\t%s // %s\n", clothes.getColor(), clothes.getType());  
 }  
 }  
 }  
  
 public static void insertCustomer(Session session) {  
 Scanner input = new Scanner(System.*in*);  
 System.*out*.println("Input a name: ");  
 String name = input.next();  
 System.*out*.println("Input a surname: ");  
 String surname = input.next();  
 System.*out*.println("Input an address: ");  
 String address = input.next();  
 System.*out*.println("Input a pass number: ");  
 int pass = Integer.*parseInt*(input.next());  
 System.*out*.println("Input a payment: ");  
 String payment = input.next();  
  
 session.beginTransaction();  
 CustomerEntity cityEntity = new CustomerEntity(name, surname, address, pass, payment);  
 session.save(cityEntity);  
 session.getTransaction().commit();  
  
 System.*out*.println("End insert customer");  
 }  
  
 public static void insertMaker(Session session) {  
 Scanner input = new Scanner(System.*in*);  
 System.*out*.println("Input company name: ");  
 String company = input.nextLine();  
 System.*out*.println("Input name: ");  
 String name = input.next();  
 System.*out*.println("Input surname: ");  
 String surname = input.next();  
 System.*out*.println("Input price: ");  
 int price = Integer.*parseInt*(input.next());  
  
 session.beginTransaction();  
 MakerEntity maker = new MakerEntity(company, name, surname, price);  
 session.save(maker);  
 session.getTransaction().commit();  
 System.*out*.println("End insert maker");  
 }  
  
 public static void insertClothes(Session session) {  
 Scanner input = new Scanner(System.*in*);  
 System.*out*.println("Input type: ");  
 String type = input.next();  
 System.*out*.println("Input amount: ");  
 int amount = Integer.*parseInt*(input.next());  
 System.*out*.println("Input color: ");  
 String color = input.next();  
 System.*out*.println("Input size: ");  
 String size = input.next();  
 System.*out*.println("Input company name: ");  
 String company = input.next();  
  
 session.beginTransaction();  
 Query query = session.createQuery(new StringBuilder().append("from MakerEntity where companyName='")  
 .append(company).append("'").toString());  
 for (Object obj : query.list()) {  
 MakerEntity maker = (MakerEntity) obj;  
 ClothesEntity clothes = new ClothesEntity(type, amount, color, size, maker);  
 session.save(clothes);  
 }  
 session.getTransaction().commit();  
 System.*out*.println("End insert clothes");  
 }  
  
 public static void updateCustomerName(Session session) {  
 Scanner input = new Scanner(System.*in*);  
 System.*out*.println("\nInput old customer name: ");  
 String name = input.nextLine();  
 System.*out*.println(name);  
 System.*out*.println("Input new customer name: ");  
 String newName = input.next();  
 int pass = 0;  
 Query query = session.createQuery(new StringBuilder().append("from CustomerEntity where firstName='")  
 .append(name).append("'").toString());  
 for (Object obj : query.list()) {  
 CustomerEntity c = (CustomerEntity) obj;  
 pass = c.getPassName();  
 }  
  
 CustomerEntity customerEntity = (CustomerEntity) session.load(CustomerEntity.class, pass);  
 if (customerEntity != null) {  
 session.beginTransaction();  
 Query new\_query = session.createQuery("update CustomerEntity set firstName=:code1 where firstName = " +  
 ":code2");  
 new\_query.setParameter("code1", newName);  
 new\_query.setParameter("code2", name);  
 int result = new\_query.executeUpdate();  
 session.getTransaction().commit();  
 System.*out*.println("End update customer name: " + result);  
 } else System.*out*.println("There is no customer with this name");  
 }  
  
 public static void AddClothesForCustomer(Session session) {  
 System.*out*.println("Give a clothes to customer --------------");  
 Scanner input = new Scanner(System.*in*);  
 System.*out*.println("Choose customer name:");  
 String name = input.next();  
 System.*out*.println("Choose clothes type:");  
 String typeCode = input.next();  
 System.*out*.println("Choose clothes color:");  
 String colorCode = input.next();  
  
 Query query = session.createQuery("from " + "CustomerEntity where firstName = :code");  
 query.setParameter("code", name);  
  
 if (!query.list().isEmpty()) {  
 CustomerEntity customerEntity = (CustomerEntity) query.list().get(0);  
 query = session  
 .createQuery("from " + "ClothesEntity where type = :type\_code and color = :color\_code");  
 query.setParameter("type\_code", typeCode);  
 query.setParameter("color\_code", colorCode);  
 if (!query.list().isEmpty()) {  
 ClothesEntity clothesEntity = (ClothesEntity) query.list().get(0);  
 session.beginTransaction();  
 customerEntity.addClothesEntity(clothesEntity);  
 session.save(customerEntity);  
 session.getTransaction().commit();  
 System.*out*.println("End insert clothes for customer");  
 } else {  
 System.*out*.println("There is no clothes with this type");  
 }  
 } else {  
 System.*out*.println("There is no customer with this name");  
 }  
  
 }  
  
 public static void deleteCustomerByName(Session session) {  
 Scanner input = new Scanner(System.*in*);  
 System.*out*.println("\nInput customer name: ");  
 String name = input.next();  
 int pass = 0;  
 Query new\_query = session.createQuery(new StringBuilder().append("from CustomerEntity where firstName='")  
 .append(name).append("'").toString());  
 for (Object obj : new\_query.list()) {  
 CustomerEntity c = (CustomerEntity) obj;  
 pass = c.getPassName();  
 }  
 CustomerEntity customerEntity = (CustomerEntity) session.load(CustomerEntity.class, pass);  
 if (customerEntity != null) {  
 session.beginTransaction();  
 Query query = session.createQuery("delete CustomerEntity where firstName=:code");  
 query.setParameter("code", name);  
 int result = query.executeUpdate();  
 session.getTransaction().commit();  
 System.*out*.println("End deleting customer by name: " + result);  
 } else System.*out*.println("There is no customer with this name");  
 }  
}

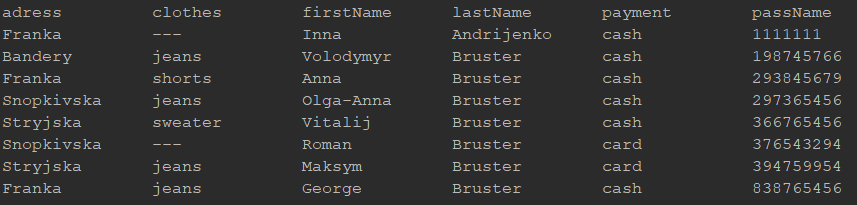
**РЕЗУЛЬТАТИ**

Вивід стикувальної таблиці:

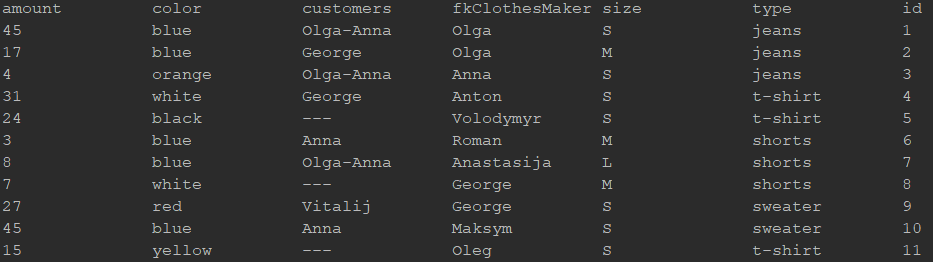


Вставка у стикувальну таблицю:

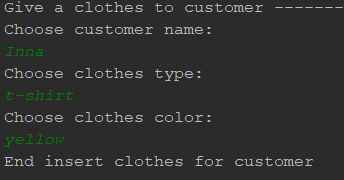
До (customer):



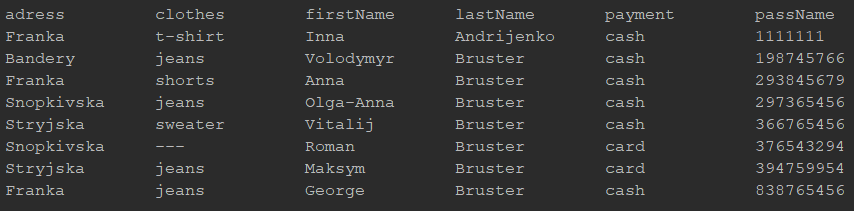
До (clothes):



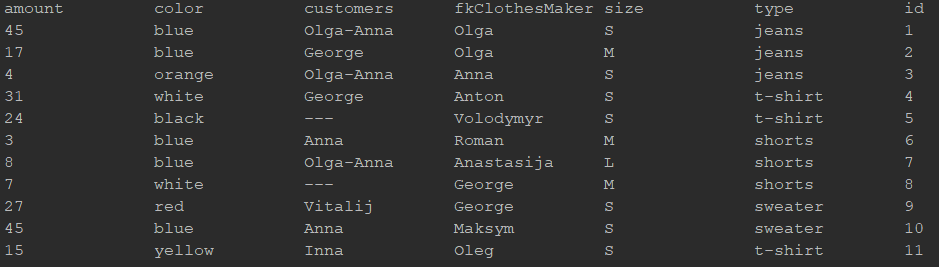
Вставка:



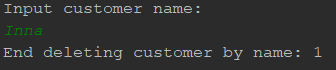
Після (customer):



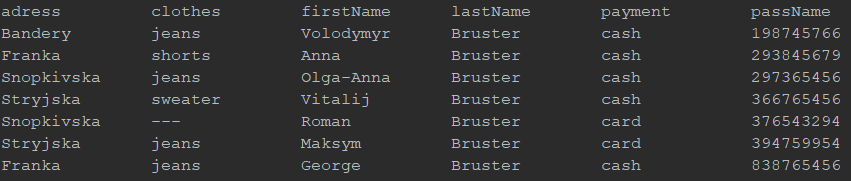
Після (clothes):



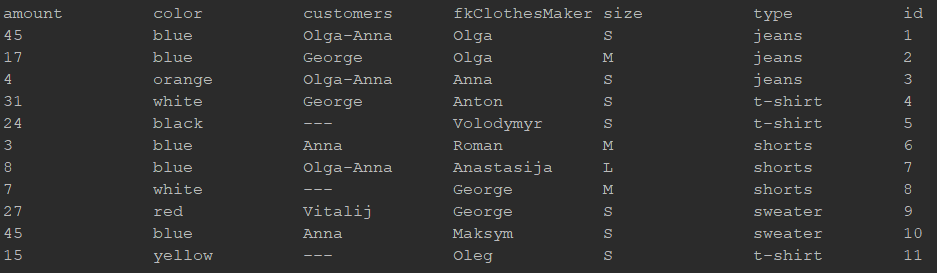
Видалення зі стикувальної таблиці:



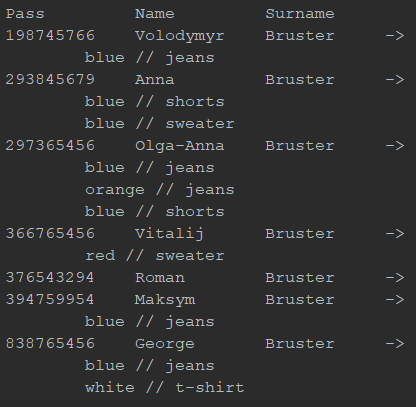
customer:



clothes:



Стикувальна таблиця:



**Висновок:** На цій лабораторній роботі я навчилася створювати CRUD-операції для роботи з базою даних MySQL за допомогою Hibernate.