**과제2**

**32200327 김경민**

**1.**

**Customer 테이블에 학생 본인의 정보를 입력하는 프로그램을 작성하고 그 결과를 보이세요.**

* CustomerNumber 가 자동 생성될 수 있도록 워크밴치에서 customers 테이블의 cuttomerNumber 속성 AI 체크

|  |
| --- |
| **코드** |
| package sklee.jdbc.sec0;  import java.sql.\*;  import java.util.Scanner;  public class InsertCustomer {  /\*\*  \* Insert a new candidate  \* @param customerName  \* @param contactLastName  \* @param contactFirstName  \* @param phone  \* @param addressLine1  \* @param addressLine2  \* @param city  \* @param state  \* @param postalCode  \* @param country  \* @param salesRepEmployeeNumber  \* @param creditLimit  \* @return  \*/  public static int insertCustomer(String customerName, String contactLastName, String contactFirstName, String phone, String addressLine1, String addressLine2,  String city, String state, String postalCode, String country, int salesRepEmployeeNumber, Double creditLimit){  // for insert a new customer  ResultSet rs = null;  int customerNumber = 0;  String sql = "INSERT INTO customers(customerName,contactLastName,contactFirstName,phone,addressLine1,addressLine2,city,state,postalCode, country, salesRepEmployeeNumber,creditLimit) "  + "VALUES(?,?,?,?,?,?,?,?,?,?,?,?) "; //수행할 쿼리  try (Connection conn = JDBC\_Util.getConnection();  PreparedStatement pstmt = conn.prepareStatement(sql, Statement.RETURN\_GENERATED\_KEYS);) { //statement 객체 생성해 다양한 메소드 사용 가능(키 자동 생성)  // set parameters for statement  pstmt.setString(1, customerName);  pstmt.setString(2, contactLastName);  pstmt.setString(3, contactFirstName);  pstmt.setString(4, phone);  pstmt.setString(5, addressLine1);  pstmt.setString(6, addressLine2);  pstmt.setString(7, city);  pstmt.setString(8, state);  pstmt.setString(9, postalCode);  pstmt.setString(10, country);  pstmt.setInt(11, salesRepEmployeeNumber);  pstmt.setDouble(12, creditLimit);  int rowAffected = pstmt.executeUpdate();  if(rowAffected == 1) //영향을 받은 행의 갯수가 1개인 경우  {  // get candidate id  rs = pstmt.getGeneratedKeys(); //sql문을 실행한 결과를 반환(key가 생성 되고 이를 반환 하는 메소드 실행)  if(rs.next()) {  customerNumber = rs.getInt(1); //insert된 행의 첫번째 인덱스의 값 가져오기  }  }  } catch (SQLException ex) {  System.out.println(ex.getMessage());  }finally {  try {  if(rs != null) rs.close();  } catch (SQLException e) {  System.out.println(e.getMessage());  }  }  return customerNumber;  }  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args){  //enter customer info  Scanner scanner = new Scanner(System.in);  String customerName = null;  String contactLastName = null;  String contactFirstName = null;  String phone = null;  String addressLine1 = null;  String addressLine2 = null;  String city = null;  String state = null;  String postalCode = null;  String country = null;  int salesRepEmployeeNumber = 0;  Double creditLimit = 0.0;  System.out.println("Please Enter customerName.");  customerName = scanner.nextLine();  System.out.println("Please Enter contactLastName.");  contactLastName = scanner.nextLine();  System.out.println("Please Enter contactFirstName.");  contactFirstName = scanner.nextLine();  System.out.println("Please Enter phone. Ex)xxx-xxxx-xxxx");  phone = scanner.nextLine();  System.out.println("Please Enter addressLine1.");  addressLine1 = scanner.nextLine();  System.out.println("Please Enter addressLine2.");  addressLine2 = scanner.nextLine();  System.out.println("Please Enter city.");  city = scanner.nextLine();  System.out.println("Please Enter state.");  state = scanner.nextLine();  System.out.println("Please Enter postalCode.");  postalCode = scanner.nextLine();  System.out.println("Please Enter country.");  country = scanner.nextLine();  System.out.println("Please Enter salesRepEmployeeNumber.");  salesRepEmployeeNumber = scanner.nextInt();  System.out.println("Please Enter creditLimit.");  creditLimit = scanner.nextDouble();  // insert a new customer  int customerNumber = insertCustomer(customerName, contactLastName, contactFirstName, phone,  addressLine1, addressLine2, city, state, postalCode, country, salesRepEmployeeNumber, creditLimit);  System.out.println(String.format("A new customer with id %d has been inserted.", customerNumber));  }  } |

|  |
| --- |
| 실행 결과 (캡쳐) |
|  |

**2.**

**본인이 원하는 제품 명을 입력하면 제품 관련 정보를 찾아 orders와 orderdetails에 입력하는 프로그램을 작성하고 그 결과를 보이세요.**

* **제품이름(productName) 입력시, 제품코드(productCode)와 권장소비자가격(MSRP) 추출 \* orderdetails 테이블의 priceEach가 상품마다 다르기 때문에 정해진 가격이 딱히 없다고 판단하고, products 테이블의 buyPrice도 판매 가격이 아닌 도매 가격이라고 판단. 따라서 priceEach는 권장소비자가격을 뜻하는 MSPR로 정함**
* **상품코드로 orders 테이블에 주문 생성, orderNumber은 1번 문제와 동일하게 자동생성 되도록 설정 바꿔줌**
* **주문 번호로 orderdetails 테이블에 주문상세 생성**
* **주문 생성 완료 후 products 테이블의 재고(quanttyInStock) update**

|  |
| --- |
| **코드** |
| package sklee.jdbc.sec0;  import java.sql.\*;  import java.text.DateFormat;  import java.text.SimpleDateFormat;  import java.util.Calendar;  import java.util.Scanner;  import java.util.Date;  import static java.lang.System.exit;  public class OrderProgram {  String productCode = null;  double msrp = 0;  int quantityInStock = 0;  public void searchProduct(String product){  //search product info  String sql = "SELECT productCode, quantityInStock, MSRP FROM products WHERE productName = ? "; //수행할 쿼리  try (Connection conn = JDBC\_Util.getConnection(); //DB 연결 객체 생성  PreparedStatement pstmt = conn.prepareStatement(sql);) { //SQL문을 실행하고 ResultSet 형식의 결과 리턴  pstmt.setString(1, product);  ResultSet rs = pstmt.executeQuery();  while (rs.next()) {  productCode = rs.getString("productCode");  quantityInStock = rs.getInt("quantityInStock");  msrp = rs.getDouble("MSRP");  if(productCode == null){  System.out.println("Product Name is invalid.");  }  if(quantityInStock == 0){  System.out.println("Sorry, We don't have stock.");  exit(0);  }  }  } catch (SQLException ex) {  System.out.println(ex.getMessage());  }  System.out.println(String.format("A product(product code : %s ) is %d stocked.",productCode, quantityInStock));  }  public int createOrder(String requiredDate,String comments,int customerNumber){  ResultSet rs = null;  int orderNumber = 0;  String sql = "INSERT INTO orders(orderDate, requiredDate, shippedDate, status, comments, customerNumber)"  + "VALUES(?,?,?,?,?,?) "; //수행할 쿼리  try (Connection conn = JDBC\_Util.getConnection(); //DB 연결 객체 생성  PreparedStatement pstmt = conn.prepareStatement(sql, Statement.RETURN\_GENERATED\_KEYS);) { //SQL문을 실행하고 ResultSet 형식의 결과 리턴  DateFormat df = new SimpleDateFormat("yyyy-MM-dd");  Calendar cal = Calendar.getInstance();  cal.setTime(new Date());  String orderDate = df.format(cal.getTime());  //set shippedDate(orderDate + 3)  cal.add(Calendar.DATE, 3);  String shippedDate = df.format(cal.getTime());  pstmt.setString(1, orderDate);  pstmt.setString(2, requiredDate);  pstmt.setString(3, shippedDate);  pstmt.setString(4, "Shipped");  pstmt.setString(5, comments);  pstmt.setInt(6, customerNumber);  int rowAffected = pstmt.executeUpdate();  if(rowAffected == 1) //영향을 받은 행의 갯수가 1개인 경우  {  // get candidate id  rs = pstmt.getGeneratedKeys();  if(rs.next()) {  orderNumber = rs.getInt(1);  }  }  } catch (SQLException ex) {  System.out.println(ex.getMessage());  }finally {  try {  if(rs != null) rs.close();  } catch (SQLException e) {  System.out.println(e.getMessage());  }  }  return orderNumber;  }  public void createOrderDetail(int orderNumber, String productCode, int quantityOrdered, double priceEach, int orderLineNumber){  ResultSet rs = null;  String sql = "INSERT INTO orderdetails(orderNumber, productCode, quantityOrdered, priceEach, orderLineNumber)"  + "VALUES(?,?,?,?,?) "; //수행할 쿼리  try (Connection conn = JDBC\_Util.getConnection(); //DB 연결 객체 생성  PreparedStatement pstmt = conn.prepareStatement(sql);) { //SQL문을 실행하고 ResultSet 형식의 결과 리턴  pstmt.setInt(1, orderNumber);  pstmt.setString(2, productCode);  pstmt.setInt(3, quantityOrdered);  pstmt.setDouble(4, priceEach);  pstmt.setInt(5, orderLineNumber);  int rowAffected = pstmt.executeUpdate();  System.out.println(String.format("%d product(s) is(are) ordered.", rowAffected));  } catch (SQLException ex) {  System.out.println(ex.getMessage());  }finally {  try {  if(rs != null) rs.close();  } catch (SQLException e) {  System.out.println(e.getMessage());  }  }  }  public void updateStock(int quantityOrdered, String productCode){  String sql = "UPDATE products "  + "SET quantityInStock = ? "  + "WHERE productCode = ? ";  try (Connection conn = JDBC\_Util.getConnection();  PreparedStatement pstmt = conn.prepareStatement(sql)){  pstmt.setInt(1, quantityInStock - quantityOrdered);  pstmt.setString(2, productCode);  int rowAffected = pstmt.executeUpdate();  System.out.println(String.format("Product %s's Stock is chaged.",productCode));  } catch (SQLException ex) {  System.out.println(ex.getMessage());  }  }  public static void main(String[] args){  OrderProgram newOrder = new OrderProgram();  Scanner scanner = new Scanner(System.in);  //Enter Product to be ordered  System.out.println("Please Enter Product Name to be ordered.");  String productName = scanner.nextLine();  //여러개인 경우  //search product code  newOrder.searchProduct(productName);  System.out.println("Do you want order? YES or NO");  if(scanner.nextLine().equals("YES")){  //Enter RequiredDate  System.out.println("Please Enter RequiredDate. Ex)xxxx-xx-xx");  String requiredDate = scanner.nextLine();  //Enter comment  System.out.println("Please Enter comment.");  String comment = scanner.nextLine();  //Enter CutomerNumber  System.out.println("Please Enter CutomerNumber.");  int cutomerNumber = scanner.nextInt();  //create order  int orderNumber = newOrder.createOrder(requiredDate, comment ,cutomerNumber);  System.out.println(String.format("A new order with id %d has been inserted.", orderNumber));  //Enter quantityOrdered  System.out.println("Please Enter quantityOrdered.");  int quantityOrdered = scanner.nextInt();  //Enter orderLineNumber  System.out.println("Please Enter orderLineNumber.");  int orderLineNumber = scanner.nextInt();  //create orderdetail  newOrder.createOrderDetail(orderNumber, newOrder.productCode, quantityOrdered, newOrder.msrp,orderLineNumber);  //update product stock  newOrder.updateStock(quantityOrdered, newOrder.productCode);  } else {  exit(0);  }  }  } |

|  |
| --- |
| 실행 결과 (캡쳐) |
| **orders 테이블 결과 화면**    **orderdetails 테이블 결과 화면**    **products 테이블 결과 화면(재고 update)** |