public class MySQLUserManageEntity implements IUserOperator {

private Connection connection;

public MySQLUserManageEntity(Connection connection) {

this.connection = connection;}

public void insert(User user) {

String sql = "INSERT INTO user(studentID,studentName, studentAge, deptID, dept, studentClass, studentAddr, studentGender) VALUES(?,?,?,?,?,?,?,?)";

try (PreparedStatement preparedStatement= connection.prepareStatement(sql)) {

preparedStatement.setInt(1, user.getStudentID());

preparedStatement.setString(2, user.getStudentName());

preparedStatement.setInt(3, user.getStudentAge());

preparedStatement.setInt(4, user.getDeptID());

preparedStatement.setString(5, user.getDept());

preparedStatement.setString(6, user.getStudentClass());

preparedStatement.setString(7, user.getStudentAddr());

preparedStatement.setString(8, user.getStudentGender());

preparedStatement.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();}}

public void delete(User user) {

String sql = "DELETE FROM user WHERE studentID = ?";

try (PreparedStatement preparedStatement = connection.prepareStatement(sql)) {

preparedStatement.setInt(1, user.getStudentID());

preparedStatement.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();}}

public void update(User user) {

String sql = "UPDATE user SET studentName = ?, studentAge = ?, deptID = ?, dept = ?, studentClass = ?, studentAddr = ?, studentGender = ? WHERE studentID = ?";

try (PreparedStatement preparedStatement = connection.prepareStatement(sql)) {

preparedStatement.setString(1, user.getStudentName());

preparedStatement.setInt(2, user.getStudentAge());

preparedStatement.setInt(3, user.getDeptID());

preparedStatement.setString(4, user.getDept());

preparedStatement.setString(5, user.getStudentClass());

preparedStatement.setString(6, user.getStudentAddr());

preparedStatement.setString(7, user.getStudentGender());

preparedStatement.setInt(8, user.getStudentID());

preparedStatement.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();}}

public User getUserByID(Integer id) {

String sql = "SELECT studentID, studentName, studentAge, deptID, dept, studentClass, studentAddr, studentGender FROM user WHERE studentID = ?";

User user = null;

try (PreparedStatement preparedStatement = connection.prepareStatement(sql)) {

preparedStatement.setInt(1, id);

ResultSet rs = preparedStatement.executeQuery();

if (rs.next()) {

user = new User(

rs.getInt("studentID"),

rs.getString("studentName"),

rs.getInt("studentAge"),

rs.getInt("deptID"),

rs.getString("dept"),

rs.getString("studentClass"),

rs.getString("studentAddr"),

rs.getString("studentGender")

);}} catch (SQLException e) {e.printStackTrace();}return user;}}

public class MySQLDepartmentManageEntity implements IDepartmentOperator {

private Connection connection;

public MySQLDepartmentManageEntity(Connection connection) {

this.connection = connection;}

public Integer insert(Department department) {

String sql = "INSERT INTO Department(name, code, manager\_id, description, status, location, phone, email) VALUES(?,?,?,?,?,?,?,?)";

try (PreparedStatement preparedStatement = connection.prepareStatement(sql, PreparedStatement.RETURN\_GENERATED\_KEYS)) {

preparedStatement.setString(1, department.getName());

preparedStatement.setString(2, department.getCode());

preparedStatement.setInt(3, department.getManager\_id());

preparedStatement.setString(4, department.getDescription());

preparedStatement.setInt(5, department.getStatus());

preparedStatement.setString(6, department.getLocation());

preparedStatement.setString(7, department.getPhone());

preparedStatement.setString(8, department.getEmail());

preparedStatement.executeUpdate();

try (ResultSet generatedKeys = preparedStatement.getGeneratedKeys()) {

if (generatedKeys.next()) {return generatedKeys.getInt(1);}}

} catch (SQLException e) {e.printStackTrace();}return null;}

public Integer delete(Department department) {

String sql = "DELETE FROM Department WHERE id = ?";

try (PreparedStatement preparedStatement = connection.prepareStatement(sql)) {

preparedStatement.setInt(1, department.getId());

return preparedStatement.executeUpdate();

} catch (SQLException e) {e.printStackTrace();}return 0;}

public Integer update(Department department) {

String sql = "UPDATE Department SET name = ?, code = ?, manager\_id = ?, description = ?, status = ?, location = ?, phone = ?, email = ? WHERE id = ?";

try (PreparedStatement preparedStatement = connection.prepareStatement(sql)) {

preparedStatement.setString(1, department.getName());

preparedStatement.setString(2, department.getCode());

preparedStatement.setInt(3, department.getManager\_id());

preparedStatement.setString(4, department.getDescription());

preparedStatement.setInt(5, department.getStatus());

preparedStatement.setString(6, department.getLocation());

preparedStatement.setString(7, department.getPhone());

preparedStatement.setString(8, department.getEmail());

preparedStatement.setInt(9, department.getId());

return preparedStatement.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();}return 0;}

public Department getDepartmentByID(String sid) {

String sql = "SELECT id, name, code, manager\_id, description, status, location, phone, email FROM Department WHERE id = ?";

Department department = null;

try(PreparedStatement preparedStatement= connection.prepareStatement(sql)) {preparedStatement.setString(1, sid);

ResultSet rs = preparedStatement.executeQuery();

if (rs.next()) {

department = new Department(

rs.getInt("id"),

rs.getString("name"),

rs.getString("code"),

rs.getInt("manager\_id"),

rs.getString("description"),

rs.getInt("status"),

rs.getString("location"),

rs.getString("phone"),

rs.getString("email"));}} catch (SQLException e) {

e.printStackTrace();}

return department;}}

public class MySQLFactory implements IFactory{

public static Connection connection=null;

public Connection getConnection(String driver, String url, String username,String password){if(connection==null){try{Class.forName(driver);

} catch (ClassNotFoundException e) {e.printStackTrace();}

try {connection = DriverManager.getConnection(url, username, password);

} catch (SQLException e) {e.printStackTrace();}return connection;}

return null;}public IUserOperator createUserManageEntity(Connection connection) {return new MySQLUserManageEntity(connection);}

public IDepartmentOperator createDepartmentManageEntity(Connection connection) {return new MySQLDepartmentManageEntity(connection);}}