University of British Columbia, Department of Computer Science

CPSC 304

2015 Summer Term 1

Project Part _2

Group Name:

_____Phocas_____

Group Members:

Name	Student Number	Unix ID	Email Address
Rock Luo	30841134	k0a9	rockthebesr@gmail.com
Ryan Quong	42183137	g4w9a	ryan_q73@hotmail.com
David Cai	32109134	n1a0b	davidcai@live.ca

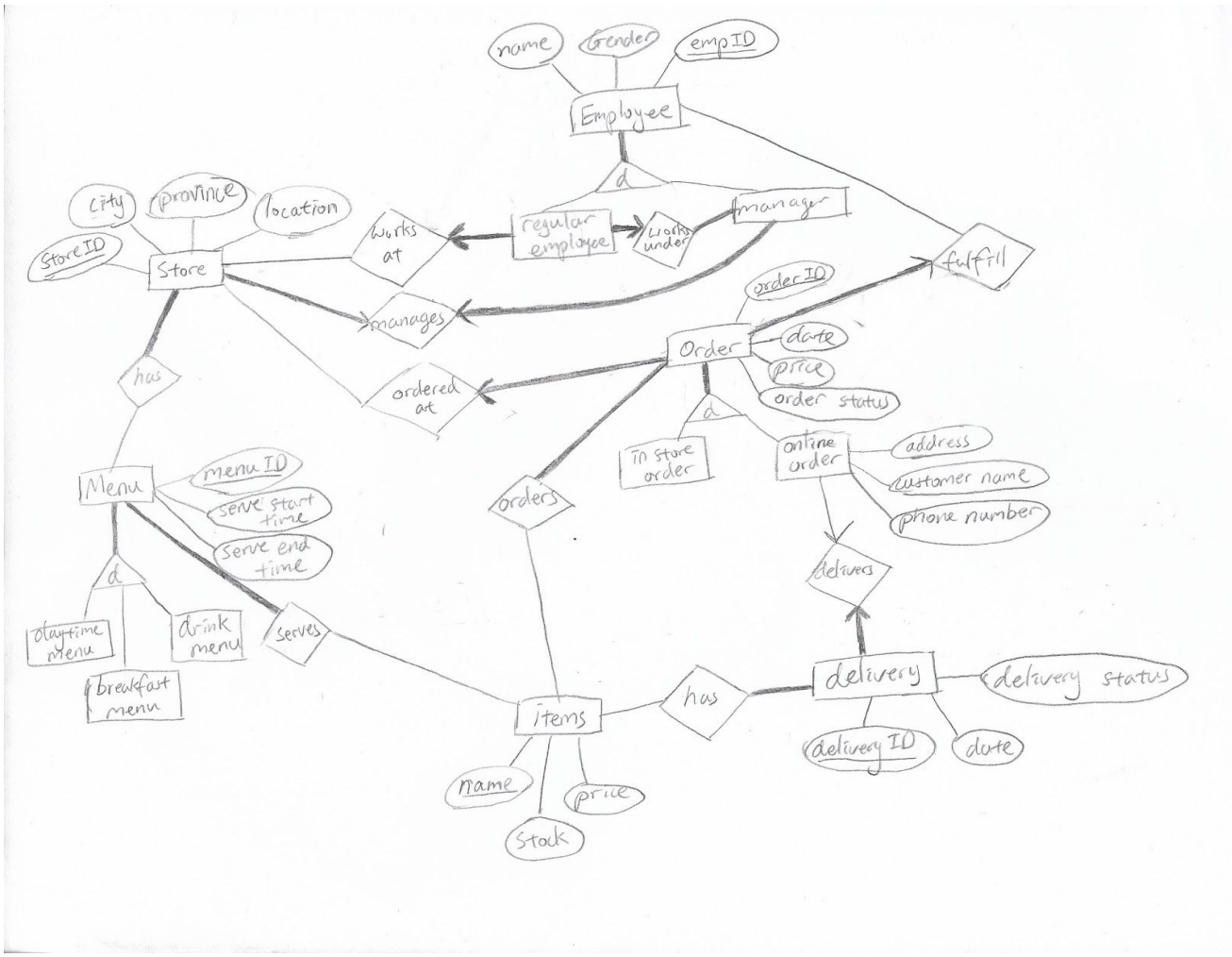
By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above.

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia.

1. Updated ER Diagram

Changes:

1. Changed “fulfill” relationship. Before it was between employee and online-order. Now it is between employee and order.
2. Underlined deliveryID in delivery
3. Underlined orderID in order
4. Moved attribtued “order status” from online order to order.



2. Set of Tables

item(<u>itemName</u>:String, stock:integer, price:Decimal(19,4))

- Represents: the entity set item
- Primary key: itemName
- Price and stock cannot be negative.

delivery(deliveryID :integer, deliveryDate: date, deliveryStatus: String)

- Represents: the entity set delivery
- Primary key: deliveryID
- Constraints: Each delivery must be associated with one online order
- Delivery status must be one of {“out on delivery” or “delivered”}
- Delivery needs to have at least one item
- Delivery needs to associate with an online-order

deliveryHasItems(**deliveryID**: integer, **itemName**: String)

- Represents: the relationship has between delivery and item
- Primary key: deliveryID, itemName
- Foreign key:
    - deliveryID references delivery(deliveryID)
    - itemName references item(itemName)

orders(**orderID**:integer, **itemName**:String)

- Represents: the relationship orders
- Primary key: orderID, itemName
- Foreign key:
    - orderID references allOrder(OrderID)
    - itemName references item(itemName) cannot be null

serves(**menuID**:integer, **itemName**:String)

- Represents: relationship serves
- Primary key: menuID, itemName
- Foreign key:
    - menuID references Menu(menuID)
    - itemName references item(itemName) cannot be null

store(storeID:integer, city: String, province: String, location: String, **empID**: int)

- Represents: the entity set store
- Primary key: storeID
- Foreign key:
    - empID references Manager(empID), empID cannot be null
- Store needs to have a set of menus
- Store needs to have a manager

storeHasMenus(**storeID**: integer, **menuID**: int)

- Represents: the relationship between store and menu
- Primary key: storeID, menuID
- Foreign key:
    - storeID references store(storeID)
    - menuID references menu(menuID)

menu(menuID: integer, serveStartTime: time, serveEndTime: time)

- Represents: the entity set menu
- Primary key: menuID
- Menu needs to serve a set of items

dayTimeMenu(**menuID**: int)

- Represents: the entity set day time menu
- Primary key: menuID
- Foreign key: menuID references menu(menuID)

breakfastMenu(**menuID**: int)

- Represents: the entity set breakfast menu
- Primary key: menuID
- Foreign key: menuID references menu(menuID)

drinkMenu(**menuID**: int)

- Represents: the entity set drink menu
- Primary key: menuID
- Foreign key: menuID references menu(menuID)

employee(empID: integer, ename: String, gender: String)

- Represents: the entity set employee
- Primary key: empID
- Gender is one of {“male”, “female”}
- Employee has to be either a regular employee or a manager

regularEmployee(**empID**: integer, **storeID**: integer, **managerID**: int)

- Represents: the worksAt-regularEmployee-worksUnder relationship set
- Primary key: empID
- Foreign key: empID references employee(empID)
    - storeID references store(storeID) cannot be null
    - managerID references manager(empID) cannot be null
- Regular employee has to work at a store
- Regular employee has to work under a manager

manager(**empID**: int)

- Represents: entity set manager
- Primary key: empID
- Foreign key:
    - empID references employee(empID)
- Manager has to have employees working under him/her
- Manager has to manage one store

manages(**empID**: integer, **storeID**: int)

- Represents: the relationship set manges

- Primary key: empID
- Foreign key:
    - empID references manager(empID)
    - storeID references store(storeID) cannot be null

allOrder(<u>orderID</u>: integer, **storeID**: integer, orderDate: date, price:Decimal(19,4), orderStatus: String, **empID**: int)

- Represents: the fulfill-order relationship set
- Primary key: orderID
- Foreign key:
    - storeID references store(storeID) cannot be null
    - empID references employee(empID) cannot be null
- Price cannot be negative
- Order has to associate with a store
- Order has to have a set of items
- Order has to be either in-store order or an online order
- orderStatus is one of {“in preparation”, “out on delivery”, “delivered”, “finished”, “cancelled”}
- order has to be fulfilled by an employee

inStoreOrder(**<u>orderID</u>**: int)

- Represents: the entity set inStoreOrder
- Primary key: orderID
- Foreign key:
    - orderID references allOrder(orderID)

onlineOrder(**<u>orderID</u>**: integer, address: String,customerName: String, phoneNumber: int)

- Represents: the onlineOrder entity set
- Primary key: orderID
- Foreign key:
    - orderID references allOrder(orderID)

delivers(**<u>deliveryID</u>**: integer, **orderID**: int)

- Represents: the delivers relationship set
- Primary key: deliveryID
- Foreign Key:
    - orderID references allOrder(orderID) cannot be null
    - deliveryID references delivery(deliveryID)

3. Functional Dependencies

item(<u>itemName</u>:String, stock:integer, price:Decimal(19,4))

- itemName -> stock, price

delivery(<u>deliveryID</u> :integer, deliveryDate: date, deliveryStatus: String)

- deliveryID -> deliveryDate, deliveryStatus

deliveryHasItems(**deliveryID**: integer, **itemName**: String)
- none

orders(**orderID**:integer, **itemName**:String)
- none

serves(**menuID**:integer, **itemName**:String)
- none

store(storeID:integer, city: String, province: String, location: String, **empID**: int)
- storeID -> city, province, location, empID
- empID -> storeID

storeHasMenus(**storeID**: integer, **menuID**: int)
- none

menu(menuID: integer, serveStartTime: time, serveEndTime: time)
- menuID -> serveStartTime, serveEndTime

dayTimeMenu(**menuID**: int)
- none

breakfastMenu(**menuID**: int)
- none

drinkMenu(**menuID**: int)
- none

employee(empID: integer, ename: String, gender: String)
- empID -> ename, gender

regularEmployee(**empID**: integer, **storeID**: integer, **managerID**: int)
- empId -> storeID, managerID

manager(**empID**: int)
- none

manages(**empID**: integer, **storeID**: int)
- empID -> storeID
- storeID -> empID

allOrder(orderID: integer, **storeID**: integer, orderDate: date, price:Decimal(19,4), **empID**: int)
- orderID -> storeID, orderDate, price, empID

inStoreOrder(**orderID**: int)

- none

onlineOrder(**orderID**: integer, address: String, customerName: String, phoneNumber: int)

- orderID -> address, customerName, phoneNumber

delivers(**deliveryID**: integer, **orderID**: int)

- deliveryID -> orderID
- orderID -> deliveryID

4. Functionalities

Make In-Store Order
User: Customer
Input: item names, storeID
Output: order ID, or “Declined”
Base Case: If items are available, an order ID will be returned.
Exception: If items are not available, “Declined” is returned.

Fulfill In-Store Order
User: Employee
Input: orderID,
Output: “Finished” or “Declined”
Base Case: If order exists in the database change its status to “Finished”
Exception: If order does not exist, return “Declined”

Cancel In-Store Order
User: Employee
Input: order ID
Output: “Order Cancelled” or “Request Denied”
Base Case: If the order is an in-store order, update the order status to “Cancelled” and return “Order Cancelled”
Exceptions: If the order is not an existing in-store order, “Request Denied” is returned.

Make Online Order
User: Customer
Input: item names, storeID
Output: order ID, or “Declined”
Base Case: If items are available, an order ID will be returned
Exceptions : If items are not available, “Declined” is returned

Fulfill Online Order
User: Employee
Input: order ID
Output: delivery ID or “Declined”

Base Case: The system checks if the order is an online order and status is “in preparation”, and then updates the order’s status to “out on delivery”, creates a new delivery, and returns the delivery ID.
Exception: If order does not exist or its status is not “in preparation”, “Declined” is returned.

Fulfill Delivery
User: Employee
Input: delivery ID
Output: “Delivered” or “Declined”
Base Case: The system checks if the delivery exists in the database. Update the delivery status to “Delivered”. Update the delivery’s related order’s status to “Delivered”.
Exception: The delivery does not exists or the delivery status is already “delivered”, return “Declined”.

Update Order Status
User: Employee
Input: order ID, new_status
Output:”updated” or “Declined”
Base Case: If order exist, update order status to new_status and return “updated”
Exception:
If order does not exist or its status is one of “Delivered”, “Finished” or “Cancelled”, “Declined” is returned.
If order is an in-store order and new_status is “Delivered”, “Declined” is returned.
If order is an online order and order status is “Out on delivery” and new_status is not “Delivered”, “Declined” is returned.

Cancel Online Purchase
User: Employee
Input: order ID
Output: “Order Cancelled” or “Request Denied”
Base Case: The system will check if the online order exists. Deletes the order. “Order Cancelled” is returned.
Exceptions: If the order is not an existing online order, “Request Denied” is returned.

Check Order Status
User: Customer or Employee
Input: order ID
Output: return online order status or “Order not found”
Base Case: The system checks if the online order exists. Returns order status.
Exception: If the order is not an existing online order, “Order not found” is returned.

Add Item
User: Manager
Input: menu ID, name of food item, cost
Output: “OK” or “Declined”

Base Case: The system will check if the menu exists. If yes, check if an item with the same name exists, if yes return “Declined”, otherwise check return “OK.”
Exception: If cost is negative then “Declined” is returned.
If menu does not exist, return “Declined”.

Add Store
User: Manager
Input:storeID, city, province and location of the new store
Output:”OK” or “Declined”
Base Case: Add the new store to the database. Return “OK.”
Exception: if a store is already exists with the same store ID then “Declined” will returned.

Add Regular Employee
User: Manager
Input: name, gender,store ID, manager ID and employee ID
Output: “OK” or “Declined”
Base Case: Add the new employee to the database. Return “OK.”
Exceptions: if an employee exists with the same ID then “Declined” will be returned
if the store ID doesn’t exist “Declined” will be returned
if the manager ID doesn’t exist “Declined” will be returned.

5. Instances Of Each Relation

ITEMNAME	STOCK	PRICE
cheeseburger	2	6.5
hashbrown	5	2
chocolate pudding	1	2
fries	10	2
chicken sandwich	3	5.5
ice cream	5	2
fish and chips	0	13.5
tuna	2	15
tomatoe soup	10	5
yogurt parfait	17	3
coke	20	1.75
sprite	20	1.75
smoothie	10	3.75
corona	5	4
guiness	7	5.75

MENUID	SERVESTARTTIME	SERVEENDTIME
1	8	11
2	8	11
3	8	11
4	8	11
5	8	11
6	8	24
7	8	24
8	8	24
9	8	24
10	8	24
11	8	24
12	8	24
13	8	24
14	8	24
15	8	24

MENUID
6
7
8
9
10

MENUID
1
2
3
4
5

MENUID
11
12
13
14
15

EMPID	ENAME	GENDER
1	James	Male
2	Lily	Female
3	Monica	Female
4	Roy	Male
5	Paul	Male
6	George	Male
7	Joanna	Female
8	David	Male
9	Ivy	Female
10	Steven	Male
11	Joy	Female

EMPID
1
2
3
4
5

STOREID	CITY	PROVINCE	LOCATION	EMPID
1	Vancouver	British Columbia	111 Granville St	1
2	Richmond	British Columbia	4567 Alexandra St	2
3	Surrey	British Columbia	240 1st Ave	3
4	Toronto	Ontario	3901 Dundas St	4
5	Toronto	Ontario	1203 Spadina St	5

EMPID	STOREID	MANAGERID
6	1	1
7	2	2
8	3	3
9	4	4
10	5	5
11	5	5

ORDERID	STOREID	ORDERDAT	PRICE	ORDERSTATUS	EMPID
1	1	16-04-22	10.5	finished	1
2	1	16-04-22	8	cancelled	1
3	2	16-04-22	5.5	finished	7
4	2	16-04-22	5.5	in preparation	7
5	3	16-04-22	6	cancelled	3
6	3	16-04-22	6	out on delivery	3
7	4	16-04-22	20	delivered	9
8	4	16-04-22	17	delivered	9
9	5	16-04-22	9.75	finished	11
10	5	16-04-22	1.75	finished	11

ORDERID
1
2
3
4
5

ORDERID	ADDRESS	CUSTOMERNAME	PHONENUMBER
6	8295 Scott Road	Simon	6045079393
7	579 Yonge Street	Claire	6473442637
8	13 Baldwin Street	Marcus	4167928858
9	120 Lombard Avenue	Sameer	6478961774
10	92 Front Street E	Ruth	4163927219

DELIVERYID	DELIVERY	DELIVERYSTATUS
1	16-04-22	delivered
2	16-04-22	delivered
3	16-04-22	delivered
4	16-04-22	delivered
5	16-04-22	delivered

DELIVERYID	ITEMNAME
1	cheeseburger
2	chocolate pudding
3	chocolate pudding
4	fish and chips
5	fish and chips

ORDERID	ITEMNAME
1	hashbrown
2	cheeseburger
3	coke
4	fries
5	tuna

MENUID	ITEMNAME
1	hashbrown
6	cheeseburger
7	fries
8	tuna
11	coke

DELIVERYID	ORDERID
1	6
2	7
3	8
4	9
5	10

6. Platform to Use

The CS Ugrad Oracle and JDBC

7. Functionalities

The views of the application will be consists of public, regular employees and managers. Public will be able to do selection on menus, stores, and items. In addition, they will also be able to look up status of order or delivery given the proper ID. They can also make online order providing their name, address, phone number and item names. Regular employees will be able to update status for deliveries and orders(including subclasses), in addition to the functionalities of the public. Managers can modify stores, items, menus, employees and all the functionalities of regular employees.

Make In-Store Order
User: Customer
Input: item names, storeID
Output: order ID, or “Declined”
Base Case: If items are available, an order ID will be returned.
Exception: If items are not available, “Declined” is returned.

Fulfill In-Store Order
User: Employee
Input: orderID,
Output: “Finished” or “Declined”
Base Case: If order exists in the database change its status to “Finished”
Exception: If order does not exist, return “Declined”

Cancel In-Store Order
User: Employee
Input: order ID
Output: “Order Cancelled” or “Request Denied”
Base Case: If the order is an in-store order, update the order status to “Cancelled” and return “Order Cancelled”
Exceptions: If the order is not an existing in-store order, “Request Denied” is returned.

Make Online Order
User: Customer
Input: item names, storeID
Output: order ID, or “Declined”
Base Case: If items are available, an order ID will be returned
Exceptions : If items are not available, “Declined” is returned

Fulfill Online Order
User: Employee
Input: order ID
Output: delivery ID or “Declined”
Base Case: The system checks if the order is an online order and status is “in preparation”, and then updates the order’s status to “out on delivery”, creates a new delivery, and returns the delivery ID.
Exception: If order does not exist or its status is not “in preparation”, “Declined” is returned.

Fulfill Delivery
User: Employee
Input: delivery ID
Output: “Delivered” or “Declined”
Base Case: The system checks if the delivery exists in the database. Update the delivery status to “Delivered”. Update the delivery’s related order’s status to “Delivered”.

Exception: The delivery does not exists or the delivery status is already “delivered”, return “Declined”.

Update Order Status
User: Employee
Input: order ID, new_status
Output:”updated” or “Declined”
Base Case: If order exist, update order status to new_status and return “updated”
Exception:
If order does not exist or its status is one of “Delivered”, “Finished” or “Cancelled”, “Declined” is returned.
If order is an in-store order and new_status is “Delivered”, “Declined” is returned.
If order is an online order and order status is “Out on delivery” and new_status is not “Delivered”, “Declined” is returned.

Cancel Online Purchase
User: Employee
Input: order ID
Output: “Order Cancelled” or “Request Denied”
Base Case: The system will check if the online order exists. Deletes the order. “Order Cancelled” is returned.
Exceptions: If the order is not an existing online order, “Request Denied” is returned.

Check Order Status
User: Customer or Employee
Input: order ID
Output: return online order status or “Order not found”
Base Case: The system checks if the online order exists. Returns order status.
Exception: If the order is not an existing online order, “Order not found” is returned.

Add Item
User: Manager
Input: menu ID, name of food item, cost
Output: “OK” or “Declined”
Base Case: The system will check if the menu exists. If yes, check if an item with the same name exists, if yes return “Declined”, otherwise check return “OK.”
Exception: If cost is negative then “Declined” is returned.
If menu does not exist, return “Declined”.

Add Store
User: Manager
Input:storeID, city, province and location of the new store
Output:”OK” or “Declined”
Base Case: Add the new store to the database. Return “OK.”
Exception: if a store is already exists with the same store ID then “Declined” will returned.

Add Regular Employee
User: Manager
Input: name, gender,store ID, manager ID and employee ID
Output: “OK” or “Declined”
Base Case: Add the new employee to the database. Return “OK.”
Exceptions: if an employee exists with the same ID then “Declined” will be returned
if the store ID doesn’t exist “Declined” will be returned
if the manager ID doesn’t exist “Declined” will be returned.

8. Division of Labour

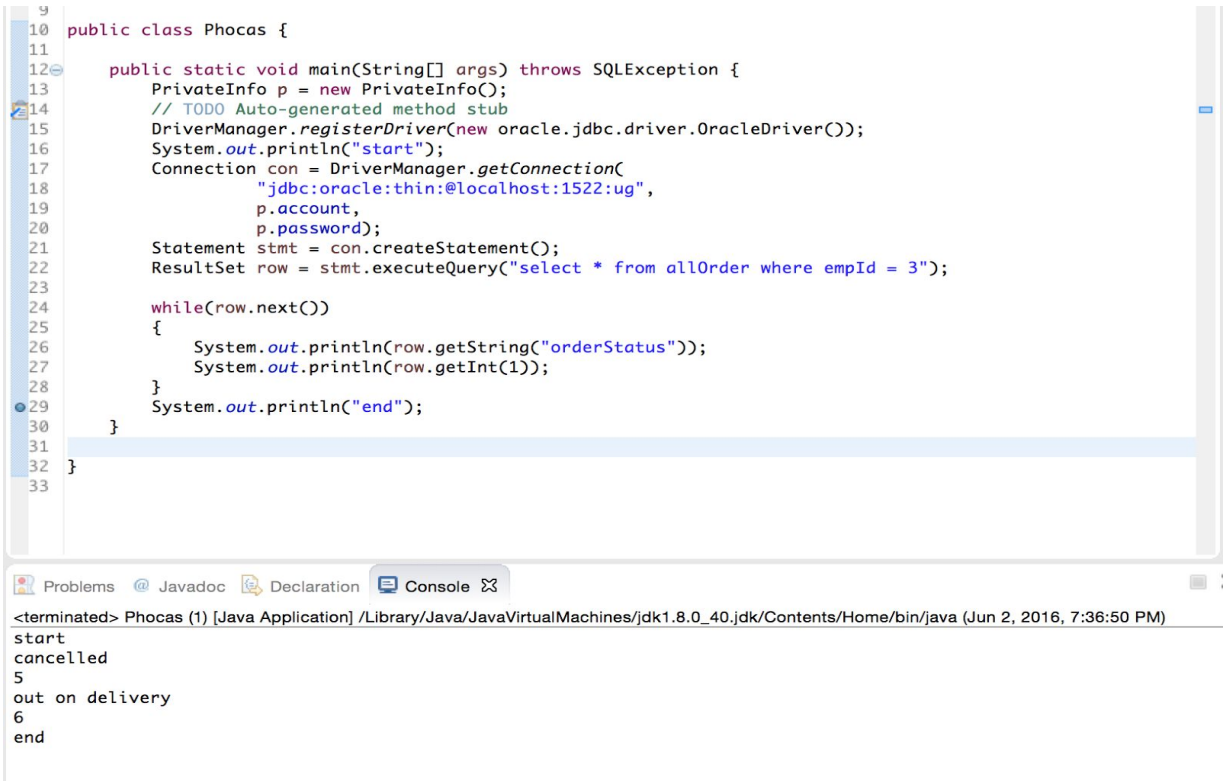
David will work on populating the database. Ryan will work on functionality. Rock will work on GUI.

9. Data for Application

We will populate the database with our own data.

10. Code that connects to database

The github page is https://github.com/rockthebesr/PhocasFoodRestaurant
Sample screenshot:



```
public class Phocas {

    public static void main(String[] args) throws SQLException {
        PrivateInfo p = new PrivateInfo();
        // TODO Auto-generated method stub
        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
        System.out.println("start");
        Connection con = DriverManager.getConnection(
                  "jdbc:oracle:thin:@localhost:1522:ug",
                  p.account,
                  p.password);
        Statement stmt = con.createStatement();
        ResultSet row = stmt.executeQuery("select * from allOrder where empId = 3");

        while(row.next())
        {
            System.out.println(row.getString("orderStatus"));
            System.out.println(row.getInt(1));
        }
        System.out.println("end");
    }

}
```

Problems Javadoc Declaration Console

```
<terminated> Phocas (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_40.jdk/Contents/Home/bin/java (Jun 2, 2016, 7:36:50 PM)
start
cancelled
5
out on delivery
6
end
```