COURSE: JAVA PROGRAMMING
DIGITAL ASSIGNMENT-1
SLOT: G1+TG1

GROUP MEMBERS
17BIT0359 – LOGESWARI S
17BCE0396 – MRINALINI SINGH

TOPIC : RESTAURANT MANAGEMENT SYSTEM

Description:

The main purpose is to **improve the performance** of the restaurant by eradicating the daily paperwork. With this system the tasks would be performed in less amount of time and more efficiently. An additional benefit of this software is that during the rush hours the load can be balanced effectively, and restaurants would perform better than usual. In addition to this, human error that occurs when performing tasks manually is also minimized and presence of queues in the system to assign tasks to chefs can reduce congestion in the kitchen. The system would also result in reduction of labor which would result in the reduction of expenses of the restaurant. Feedback module would help the restaurant check for how well they are performing, and monthly/yearly figures can be checked by the billing module to see the trends in sales and profits. These benefits can potentially result in generation of more revenues for the restaurant.

PURPOSE:

Online Restaurant management system is the system for manage the restaurant business. After successful login the customer can access the menu page with the items listed according to the desired time. The main point of developing this system is to help restaurant administrator manage the restaurant business and help customer for online ordering and reserve table. In proposed system user can search for a menu according to his choice i.e. according to price range and category of food and later he can order a meal.

The project is developing because; many restaurants have a lot difficult to manage the business such as customer ordering and reservation table. If the customer book an order and later wants to cancel the order, he is permitted to do this only within a specific time period. By using manual customer ordering is difficult to waiter keep the correct customer information and maybe loss the customer information. The customer is also given with the facility to view the status of the order and if the order is ready then he can go and get it.

So, online restaurant management system will develop to help the restaurant administrator to manage restaurant management and for customer make their online ordering and reservation table. At Management side, initially the staff member has to login, and according to his designation the privileges are set. Other than that, this project is to upgrade the manual system and make the business easily to access and systematic. If the staff member is a cook, then he is allowed to edit only the order items status, indicating which menu items he has prepared.

LOGO:

README:

Resutaurant Management System (RMS)
Execute

website :- https://projectworlds.in

Login
You can use test data for the first time. You can add new staff when you log in as manager.
Manager
- ID:1000 Password:Java
- ID:5555 Password:kazukazu

Staff
* ID:1111 Password:password
* ID:3333 Password:logeswari
(Modifing the data file directoly may make problem.)

Show menu
You can see all menu items by clicking ALL button, and items in particular categories by clicking Drink, Alcohol, Main, or Dessert button.

Taking order
Create new order
1. Click "Show menu" button on the left
2. Click "New" button to create new order
3. Select adding items by clicking from the menu list on the right side.
4. Enter quantity and click "Add" button on the left side. (If quantity is empty, one item will be added)
5. You can delete ordered item from the order detail by clicking "Delete" button

Edit order
1. Click "Show menu" button on the left
2. Select the order from the order list to edit
3. Click "Edit" button
4. You can add, delete ordered items

Close or Cancel order
1. Select the order from the order list

2. Click "Close" button or "Cancel" button
3. The order closed or canceled cannot edit

Manage Employees (Manager only)
Add new staff
1. Click "Manage Employees" Button on the left
2. Click "New" button
3. Fill in all information and click OK

###Edit staff
1. Click "Manage Employees" Button on the left
2. Select a staff from the employees list
3. Click "Edit" button
4. Fill in all information and click OK

###Delete staff
1. Click "Manage Employees" Button on the left
2. Select a staff from the employees list
3. Click "Delete" button

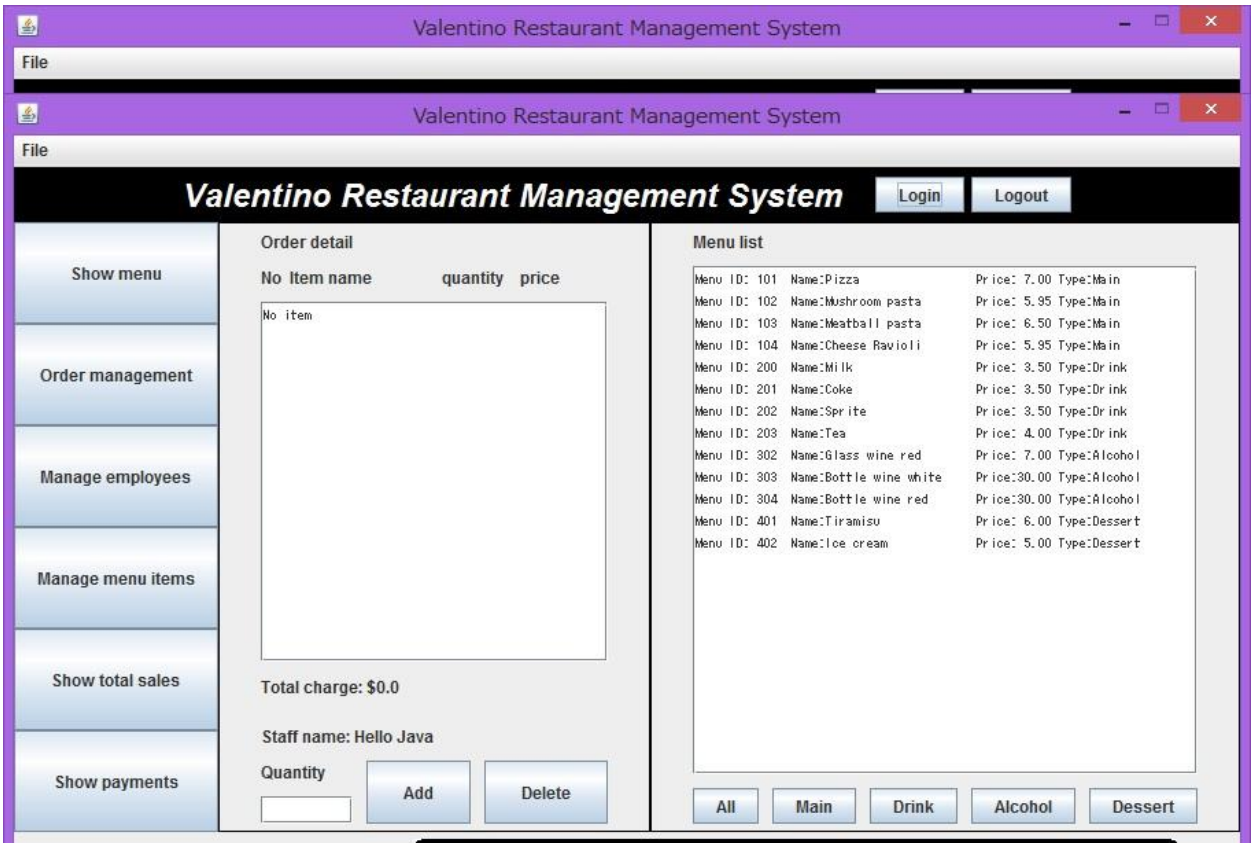
##Manage Menu Items (Manager only)
###Add new item
1. Click "Manage menu items" Button on the left
2. Click "Add new menu item" button
3. Fill in all information and click OK

###Edit menu item
1. Click "Manage menu items" Button on the left
2. Select a menu item from the menu list
3. Click "Edit menu item" button
4. Fill in all information and click OK

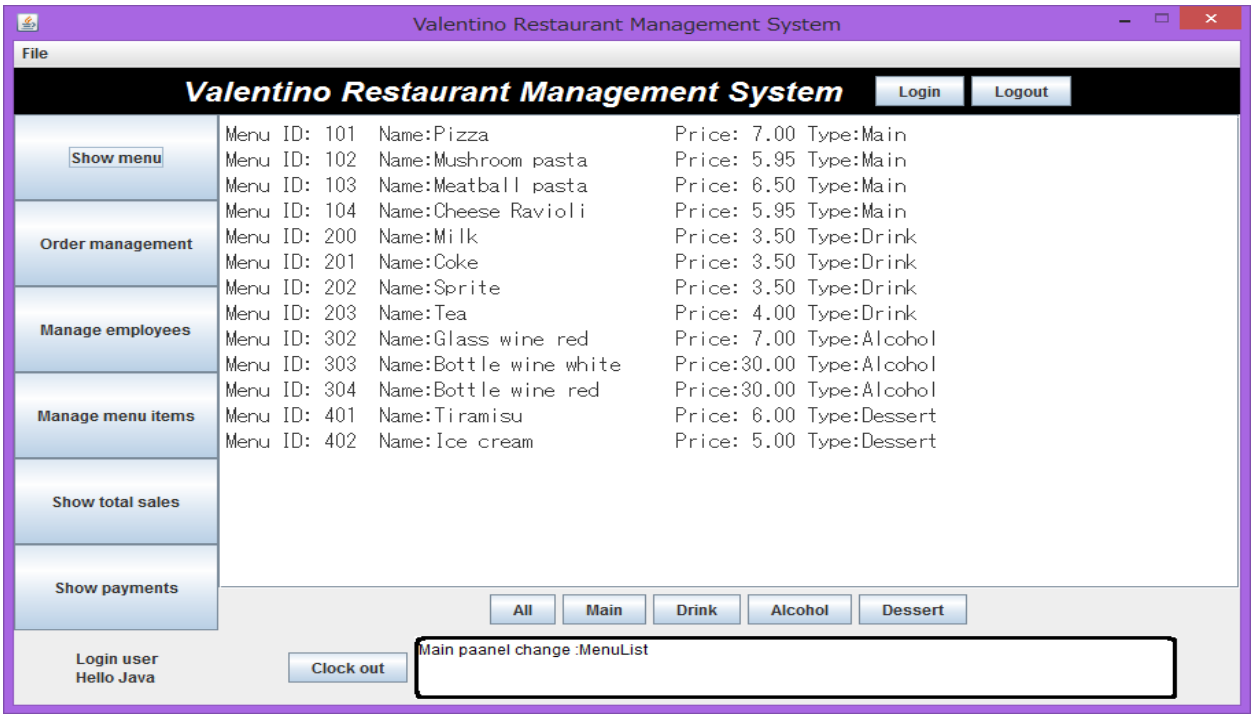
###Delete menu item
1. Click "Manage menu items" Button on the left
2. Select a menu item from the menu list
3. Click "Delete menu item" button

##About payments
* When you log in, the system automaticaly set start working time.
* Clock out button will set finish working time of the person currently logged in.
* Manager can make staff clocked out via manage employees, by selecting staff and clicking Clock out button.
* You can see a payment details for a day by clicking "Show payment" button on the left

ORDER SCREENSHOT:



MENU SCREENSHOT:



CODE:

```
import java.util.*;
import java.io.*;
import java.text.*;

public class Controller
{
   //define scene
   public final static int SCENE_MAIN_MENU = 0;    //main menu
   public final static int SCENE_LOGIN = 1;        //login
   public final static int SCENE_LOGOUT = 2;
   public final static int SCENE_MENU_LIST = 3;
   public final static int SCENE_ORDER = 4;
   public final static int SCENE_EMPLOYEE_LIST = 5;
   public final static int SCENE_EDIT_EMPLOYEE = 6;
   public final static int SCENE_EDIT_MENU_ITEM = 7;
   public final static int SCENE_GENERATE_REPORTS = 8;

   //define user type
   public final static int USER_ANONYMOUS = 0;
   public final static int USER_EMPLOYEE = 1;
   public final static int USER_MANAGER = 2;

   private UserInterface cView; //Reference of userinterface
   private Database    cDatabase;

   //parameter
   private int scene;
   private int state;  //normally "0", if something happen (ex. quit program) this have some value.
   private int userType;
   private int currentUserID;
   private String currentUserName;
   private String    todaysDate;

   public Controller()
   {
      this.cDatabase = new Database();
      this.cView = new UserInterface(this.cDatabase);
      this.scene = SCENE_MAIN_MENU;
      this.userType = USER_ANONYMOUS;
      this.currentUserID = 0;
      this.currentUserName = "";

      //get todays date
      Date date = new Date();
```

```
    SimpleDateFormat stf = new SimpleDateFormat("yyyy/MM/dd");
    todaysDate = stf.format(date);
    cView.setTodaysDate(todaysDate);

    try
    {
       cDatabase.loadFiles();
       this.state = 0;
    }
    catch(DatabaseException de)
    {
       this.state = -1;
       printErrorMessageToView(de.getErrMessage());
    }
}

/************************************************************
 * Main menu
 ************************************************************/
//-----------------------------------------------------------
// Select number from mein menu
//-----------------------------------------------------------
private void selectMenu()
{
    cView.showMainMenu( userType);

    int selection = 0;
    while( selection == 0 && this.state == 0)
    {
       try
       {
          printMessageToView("Please make a selection:");
          String key = cView.userInput();

          if(key.equalsIgnoreCase("Q"))   //quit program
          {
             printMessageToView("Are you sure to quit program? (Y:YES)");
             key = cView.userInput();
             if (key.equalsIgnoreCase("Y")) {
                this.state = 1; //If state > 0, program will finish
             }
             else
             {
                //reflesh view
                cView.showMainMenu(userType);
             }
```

```
            }
            else if(key.equalsIgnoreCase("F")) //clock out
            {
               printMessageToView("Are you sure to clock out? (Y:YES)");
               key = cView.userInput();
               if (key.equalsIgnoreCase("Y")) {
                  Staff rStaff = cDatabase.findStaffByID(currentUserID);
                  if(rStaff.getWorkState() == Staff.WORKSTATE_FINISH)
                  {
                     printMessageToView("You already clocked out.");
                  }
                  else
                  {
                     rStaff.clockOut();
                     printMessageToView("Thanks for your hard work!!");
                  }
               }
               pause(3);
               //reflesh view
               cView.showMainMenu(userType);
            }
            else
            {
               selection = Integer.parseInt(key);
               if( selectionCheck( selection))
               {
                  this.scene = selection;
               }
               else
               {
                  //reflesh view
                  cView.showMainMenu(userType);

               }
            }
         }
         catch(Exception e)
         {
            //String errMessage = e.toString()  + ":" + e.getMessage();
            //printMessageToView(errMessage);
         }
      }
   }

   public void mainLoop()
   {
```

```
    while( state == 0)
    {
       switch( this.scene)
       {
          case SCENE_MAIN_MENU:
            selectMenu();
            break;
          case SCENE_LOGIN:
             userLogin();
             break;
          case SCENE_LOGOUT:
             userLogout();
             break;
          case SCENE_MENU_LIST:
             showMenuList();
             break;
          case SCENE_ORDER:
             selectOrderMenu();
             break;
          case SCENE_EMPLOYEE_LIST:
             showStaffList();
             break;
          case SCENE_EDIT_EMPLOYEE:
             chooseEditStaffMode();
             break;
          case SCENE_EDIT_MENU_ITEM:
             chooseEditMenuItemMode();
             break;
          case SCENE_GENERATE_REPORTS:
             //generateSalesReports();
             generateReports();
             break;
          default:
             this.scene = SCENE_MAIN_MENU;
             break;
       }
       if(state == -1) //error
          printErrorMessageToView("Error");
    }

    //finish program
    cView.finish();
}
//-------------------------------------------------------
// Check if the number selected is appropriate and
// if the user is eligible to operate
```

```
//-------------------------------------------------------
private boolean selectionCheck(int selection)
{
   boolean result = true;
   switch(userType)
   {
      case USER_ANONYMOUS:
      if( selection <= 0 || SCENE_LOGIN < selection)
         result = false;
      break;
      case USER_EMPLOYEE:
      if( selection <= 0 || SCENE_ORDER < selection)
         result = false;
      break;
      case USER_MANAGER:
      if( selection <= 0 || SCENE_GENERATE_REPORTS < selection)
         result = false;
      break;
   }
   return result;
}

/*************************************************************
 * Login mode
 *************************************************************/
private void userLogin()
{
   cView.loginView();
   printMessageToView("Login as manager? (Y/N)");

   String key = cView.userInput();
   if( key.equalsIgnoreCase("Q"))// back to main menu
   {
      scene = SCENE_MAIN_MENU;
      return;
   }

   while (!key.equalsIgnoreCase("Y") && !key.equalsIgnoreCase("N"))
   {
      printMessageToView("Please enter 'Y' or 'N'\nLogin as manager? (Y/N)");
      key = cView.userInput();
   }

   if (key.equalsIgnoreCase("Y")) {
      loginCheck(true);      //search manager data
   } else if (key.equalsIgnoreCase("N")) {
```

```
        loginCheck(false);    //search employee data
    }
    // back to main menu
    scene = SCENE_MAIN_MENU;
}

//---------------------------------------------------------
// Find user
//---------------------------------------------------------
private void loginCheck( boolean isManager)
{
    String searchClassName;
    int     inputID = 0;
    String  iuputPassword = "";
    String  key = "";

    printMessageToView("Enter your ID.");
    while(inputID == 0)
    {
        key = cView.userInput();
        try{
            inputID = Integer.parseInt(key);
        }
        catch(Exception e)
        {
            printMessageToView("Only number is accepted.\nEnter your ID.");
        }
    }
    printMessageToView("Enter your password.");
    iuputPassword = cView.userInput();

    //---------search user----------
    Staff   rStaff = cDatabase.findStaffByID(inputID);

    if(isManager)   searchClassName = "Manager";
    else            searchClassName = "Employee";

    if( rStaff != null)//User data is found
    {
        //Search only particular target(Manager or Employee)
        if( rStaff.getClass().getName().equalsIgnoreCase(searchClassName))
        {
            if(rStaff.getPassword().equals(iuputPassword))
            {
                printMessageToView("Login successful!!");
                if(rStaff.getWorkState() == 0)  //Not clocked in yet
```

```
            {
               rStaff.clockIn();
            }
            if(isManager)   userType = USER_MANAGER;
            else            userType = USER_EMPLOYEE;
            currentUserID = inputID;
            currentUserName = rStaff.getFullName();
            cView.setUserName(currentUserName);  //show user name on the view
         }
         else
         {
            printMessageToView("Password unmatching.");
         }
      }
      else    //ID is found but type(Manager or Employee) is umnatching
      {
         printMessageToView("Not found.");
      }
   }
   else
      printMessageToView("Not found.");


   pause(2);
}
//---------------------------------------------------------
// Logout (Set state as Anonymous)
//---------------------------------------------------------
private void userLogout()
{
   //cView.loginView();
   printMessageToView("Are you sure to log out? (YES:y)");

   String key = cView.userInput();
   if(key.equalsIgnoreCase("Y"))
   {
      userType = USER_ANONYMOUS;
      currentUserID = 0;
      currentUserName = "";
      cView.setUserName(currentUserName);
   }
   scene = SCENE_MAIN_MENU;
}
/*************************************************************
 * Edit employee mode
 *************************************************************/
```

```
 //---------------------------------------------------------
// Choose edit mode (1:Add 2:Update 3:Delete)
//---------------------------------------------------------
private void chooseEditStaffMode()
{
   String  key;
   int     inputNumber = 0;

   cView.staffManagementView();
   printMessageToView("Choose number:");
   key = cView.userInput();

   if(key.equalsIgnoreCase("Q"))   //Back to main menu
   {
      scene = SCENE_MAIN_MENU;
      return;
   }

   while(inputNumber == 0)
   {
      try
      {
         inputNumber = Integer.parseInt(key);
         switch(inputNumber)
         {
            case 1: //add new employee
               addNewStaff();
            break;
            case 2:
               updateStaff();
            break;
            case 3:
               deleteStaff();
            break;
            default:
               printMessageToView("Choose 1 to 3:");
               key = cView.userInput();
            break;
         }
      }
      catch(Exception e)
      {
         printMessageToView("Choose 1 to 3:");
         key = cView.userInput();
      }
   }
```

```
}
//-----------------------------------------------------------
// Add new staff
//-----------------------------------------------------------
private void addNewStaff()
{
   int newID=0;
   String newFirstName;
   String newLastName;
   String newPassword;
   String key;

   boolean done = false;
   //-------------------- loop until new staff is added or enter "Q" ----------
   while(!done)
   {
      cView.addNewStaffView();
      newID = generateID();
      if (newID == 0)
      {
         //back to mein menu
         scene = SCENE_MAIN_MENU;
         return;
      }

      printMessageToView("Enter first name:");
      newFirstName = cView.userInput();
      printMessageToView("Enter last name:");
      newLastName = cView.userInput();
      printMessageToView("Enter password:");
      newPassword = cView.userInput();

      printMessageToView("Is the staff manager?(Y/N)");
      key = cView.userInput();
      int staffType = 0;  //1:manager 2:employee

      while(staffType == 0)
      {
         if(key.equalsIgnoreCase("Y"))
         {
            staffType = 1;
            break;
         }
         else if(key.equalsIgnoreCase("N"))
         {
            staffType = 2;
```

```
                break;
            }
            else
            {
                printMessageToView("Please enter 'Y' or 'N'");
                key = cView.userInput();
            }
        }
        //Check all input
        printMessageToView("-------------------------------------");
        printMessageToView("NewID:" + newID);
        printMessageToView("New staff name:" + newFirstName + " " + newLastName);
        printMessageToView("Password:" + newPassword);
        switch(staffType)
        {
            case 1:
            printMessageToView("The staff will be added as manager.");
            break;
            case 2:
            printMessageToView("The staff will be added as employee.");
            break;
        }

        printMessageToView("\nOK? (Y:yes)");
        key = cView.userInput();

        if(key.equalsIgnoreCase("Y"))
        {
            boolean isManager = false;
            if(staffType == 1)
                isManager = true;
            try
            {
                cDatabase.addStaff(newID, newLastName, newFirstName, newPassword,
isManager);
                printMessageToView("New staff information is added.");
                done = true;
            }
            catch(DatabaseException dbe)
            {
                printErrorMessageToView(dbe.getErrMessage());
                pause(2);
            }
        }
    }
    //---------------- Staff is added (loop end)----------------------------
```

```
        printMessageToView("Please enter something to exit");
        key = cView.userInput();
        scene = SCENE_MAIN_MENU;
    }

    //---------------------------------------------------------
    // Make and check new ID (used by addEmployee method only)
    //---------------------------------------------------------
    private int generateID()
    {
        int newID = 0;
        String key;

        printMessageToView("Choose user ID for new staff:");
        key = cView.userInput();

        while(newID == 0)
        {
            //Back to main menu
            if(key.equalsIgnoreCase("Q"))
                return 0;
            try
            {
                newID = Integer.parseInt(key);
                if(newID > 9999)
                {
                    printMessageToView( "Please enter less than 10000");
                    key = cView.userInput();
                    newID = 0;
                }
                else
                {
                    //Check if there is same ID
                    Staff   rStaff = cDatabase.findStaffByID(newID);
                    //if(found)
                    if(rStaff != null)
                    {
                        printMessageToView( "ID:" + newID + "is already used by " +
rStaff.getFirstName() + " "
                                + rStaff.getLastName() + ".");
                        printMessageToView("Please try another number:");
                        key = cView.userInput();
                        newID = 0;
                    }
                }
```

```
        }
        catch(Exception e)
        {
            printMessageToView("Please enter valid integer.");
            key = cView.userInput();
        }

    }
    return newID;
}

//-----------------------------------------------------------
// Update staff info
//-----------------------------------------------------------
private void updateStaff()
{
    String  key;
    int     inputNumber = 0;
    Staff   rStaff = null;
    boolean done = false;

    cView.showStaffList();

    //------------- Input ID check ----------------
    while(inputNumber == 0)
    {
        printMessageToView("Choose user ID to edit:");
        key = cView.userInput();

        if(key.equalsIgnoreCase("Q"))
        {
            //scene = SCENE_MAIN_MENU;
            printMessageToView("Transaction is canceled.");
            pause(2);
            return;
        }

        try
        {
            //findUser
            inputNumber = Integer.parseInt(key);
            rStaff = cDatabase.findStaffByID(inputNumber);

            if(rStaff == null)
            {
                inputNumber = 0;
```

```
            printErrorMessageToView("ID is not found.");
        }
    }
    catch(Exception e)
    {
        printErrorMessageToView("ID must be valid number.");
    }
}
//------------- Input ID check end ----------------

//------------- Choose update staff menu ----------------
cView.updateStaffView(rStaff);
inputNumber = 0;

while(inputNumber == 0)
{
    key = cView.userInput();

    if(key.equalsIgnoreCase("Q"))
    {
        printMessageToView("Transaction is canceled.");
        pause(2);
        return;
    }

    try{
        inputNumber = Integer.parseInt(key);
        if(inputNumber < 1 || 5 < inputNumber)
        {
            inputNumber = 0;
            printMessageToView("Input 1 to 5");
        }
    }
    catch(Exception e)
    {
        printMessageToView("Input valid integer:");
    }
}
//------------- Choose update staff menu End----------------

 //------------- Edit Staff data ----------------
DateFormat sdf = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
while(!done){
   cView.clearScreen();
   cView.showStaffData(rStaff);
   try
```

```
        {
            switch(inputNumber)
            {
                case 1: //edit last name
                    printMessageToView("Input new last name:");
                    key = cView.userInput();
                    //rStaff.setLastName(key);
                    cDatabase.editStaffData(rStaff, cDatabase.EDIT_LAST_NAME, key);
                    cView.showStaffData(rStaff);
                    printMessageToView("Please enter something to exit");
                    key = cView.userInput();
                    done = true;
                break;
                case 2: //edit first name
                    printMessageToView("Input new first name:");
                    key = cView.userInput();
                    //rStaff.setFirstName(key);
                    cDatabase.editStaffData(rStaff, cDatabase.EDIT_FIRST_NAME, key);
                    cView.showStaffData(rStaff);
                    printMessageToView("Please enter something to exit");
                    key = cView.userInput();
                    done = true;
                break;
                case 3:// Forth clock out
                    byte state = rStaff.getWorkState();
                    switch(state)
                    {
                        case Staff.WORKSTATE_ACTIVE:
                            rStaff.clockOut();
                            printMessageToView("Staff:" + rStaff.getFullName() + " has been clocked
out.");
                            pause(2);

                            break;
                        case Staff.WORKSTATE_FINISH:
                            printErrorMessageToView("Staff:" + rStaff.getFullName() + " already
clocked out.");
                            pause(3);
                            break;
                       default:
                            printErrorMessageToView("Staff:" + rStaff.getFullName() + "has not been
on work today.");
                            pause(3);
                            break;
                    }
                    done = true;
```

```
                break;
                case 4://change start time
                   if(rStaff.getWorkState() == 0)
                   {
                      printErrorMessageToView("You can not change start time of the staff not
working.");
                      pause(3);
                      return;
                   }

                   printMessageToView("Enter new start time (HH:mm)");
                   key = cView.userInput();
                   if(key.equalsIgnoreCase("Q"))
                   {
                      printMessageToView("Transaction is canceled.");
                      pause(2);
                      return;
                   }
                   key = todaysDate + " " + key + ":00"; //YYYY/MM/DD HH:mm:ss
                   try
                   {
                      Date newTime = sdf.parse(key);
                      if(rStaff.changeStartTime(newTime))
                      {
                         printMessageToView("Start time has been changed.");
                         printMessageToView("Please enter something to exit");
                         key = cView.userInput();
                         done = true;
                      }
                      else
                      {
                         printErrorMessageToView("changeStartTime error");
                         pause(3);
                      }
                   }
                   catch(ParseException pe)
                   {
                      printErrorMessageToView("Parse error");
                      printMessageToView("Follow the format 'HH:mm' (ex: 16:30)");
                      pause(3);
                   }
                break;
                case 5://change finish time
                   if(rStaff.getWorkState() != Staff.WORKSTATE_FINISH)
                   {
```

```
                    printErrorMessageToView("You can not change finish time of the staff not
working.");
                    pause(3);
                    return;
                }

                printMessageToView("Enter new finish time (HH:mm)");
                key = cView.userInput();
                if(key.equalsIgnoreCase("Q"))
                {
                    printMessageToView("Transaction is canceled.");
                    pause(2);
                    return;
                }
                key = todaysDate + " " + key + ":00";   //YYYY/MM/DD HH:mm:ss

                try
                {
                    Date newTime = sdf.parse(key);
                    if(rStaff.changeFinishTime(newTime))
                    {
                        printMessageToView("Finish time has been changed.");
                        printMessageToView("Please enter something to exit");
                        key = cView.userInput();
                        done = true;
                    }
                    else
                    {
                        printErrorMessageToView("changeFinishTime error");
                        pause(3);
                    }
                }
                catch(ParseException pe)
                {
                    printErrorMessageToView("Parse error");
                    printMessageToView("Follow the format 'HH:mm' (ex: 16:30)");
                    pause(3);
                }
            break;
            default:
                printErrorMessageToView("This line must not be used!!");
                printErrorMessageToView("Check Controller class");
                pause(2);
            break;
        }
    }
```

```
        catch(DatabaseException dbe)
        {
            printErrorMessageToView(dbe.getErrMessage());
            pause(3);
        }
        //------------- End of edit Staff data ----------------
    }
    //----- end loop(loop until done is true) ---------

    if(rStaff.getID() == currentUserID)
    {
        currentUserName = rStaff.getFullName();
        cView.setUserName(currentUserName);
    }
}

//-----------------------------------------------------------
// Delete staff
//-----------------------------------------------------------
private void deleteStaff()
{
    String  key;
    int     inputNumber = 0;
    Staff   rStaff = null;

    cView.showStaffList();
    printMessageToView("Choose user ID to delete:");
    key = cView.userInput();

    if(key.equalsIgnoreCase("Q"))   //Back to main menu
    {
        scene = SCENE_MAIN_MENU;
        return;
    }

     while(inputNumber == 0)
    {
        try
        {
            //findUser
            inputNumber = Integer.parseInt(key);
            rStaff = cDatabase.findStaffByID(inputNumber);

            if(rStaff == null)
            {
                printMessageToView("ID is not found.");
```

```
            pause(2);
            // back to main menu
            scene = SCENE_MAIN_MENU;
            return;
         }

         printMessageToView("Staff ID:" + rStaff.getID() + " Name:" + rStaff.getFirstName()
                  + " " + rStaff.getLastName() + "will be deleted. OK? (YES:y)");

         key = cView.userInput();
         if(!key.equalsIgnoreCase("Y"))
         {
            printMessageToView("The transaction is canceled.");
            pause(2);
            scene = SCENE_MAIN_MENU;
            return;
         }

         cDatabase.deleteStaff(rStaff);
         /*if(rStaff.getClass().getName().equalsIgnoreCase("Manager"))
            cDatabase.updateStaffFile(true);//update manager file
         else
            cDatabase.updateStaffFile(false);//update employee file*/

         printMessageToView("Deleted.");
         pause(2);
      }
      catch(Exception e)
      {
         printErrorMessageToView("ID must be valid number. Input again:");
         key = cView.userInput();
      }
   }
}
 /************************************************************
 * Order mode
 ************************************************************/
//---------------------------------------------------------
// Choose order mode (1:Create order 2:Close order 3:Cancel order)
//---------------------------------------------------------
private void selectOrderMenu()
{
   String  key;
   int     inputNumber = 0;

   Staff rStaff = cDatabase.findStaffByID(currentUserID);
```

```
if(rStaff.getWorkState() == Staff.WORKSTATE_FINISH)
{
   printErrorMessageToView("You already clocked out.");
   pause(2);
   scene = SCENE_MAIN_MENU;
   return;
}

while(inputNumber == 0)
{
   cView.showOrderMenu();
   printMessageToView("Choose number:");
   key = cView.userInput();

   if(key.equalsIgnoreCase("Q"))   //Back to main menu
   {
      scene = SCENE_MAIN_MENU;
      return;
   }

   try
   {
      inputNumber = Integer.parseInt(key);
      if(inputNumber < 1 || 5 < inputNumber)
      {
         inputNumber = 0;
         printErrorMessageToView("Choose 1 to 5:");
         pause(2);
      }
   }
   catch(Exception e)
   {
      printErrorMessageToView("Choose 1 to 5:");
      pause(2);
   }
}

switch(inputNumber)
{
   case 1: //Create
      createOrder();
   break;
   case 2: //Update
      updateOrder();
   break;
   case 3://Close
```

```
            closeOrder();
         break;
         case 4://Cancel
            cancelOrder();
         break;
         case 5://Show order list
            showOrderList();
            /*cView.showOrderList();
            //printMessageToView("Please enter something to exit.");
            //key = cView.userInput();
            printMessageToView("Enter order ID to display detail. (Q:quit)");
            key = cView.userInput();
            if(!key.equalsIgnoreCase("Q"))
            {
               try
               {
               }

            }*/
         break;
         default:
            printErrorMessageToView("This line must not be exexuted!! Check program of
selectOrderMenu()");
            key = cView.userInput();
         break;
      }
   }

   private void showOrderList()
   {
      boolean done = false;
      String key;

      while(!done)
      {
         cView.showOrderList();
         printMessageToView("Enter order ID to display detail. (Q:quit)");
         key = cView.userInput();
         if(key.equalsIgnoreCase("Q"))   //Exit
         {
            return;
         }

         try{
            int ID = Integer.parseInt(key);
            Order rOrder = cDatabase.findOrderByID(ID);
```

```
            if( rOrder == null)
            {
               printErrorMessageToView("Not found.");
               pause(2);
            }
            else
            {
               cView.clearScreen();
               cView.showOrderDetail(ID);
               printMessageToView("Please enter something to exit.");
               key = cView.userInput();
               done = true;
            }
         }
         catch(Exception e)
         {
            printErrorMessageToView("Enter valid integer.");
            pause(2);
         }
      }
   }

   private void createOrder()
   {
      int newOrderID = cDatabase.addOrder(currentUserID, currentUserName);
      editOrderItem(newOrderID);
   }

   private void updateOrder()
   {
      cView.showOrderList();
      int updateOrderID = findOrder();
      if( updateOrderID == -1)
         return;

      Order rOrder = cDatabase.findOrderByID(updateOrderID);
      if( currentUserID != rOrder.getStaffID())
      {
         printErrorMessageToView("You are not eligible to update the order.");
         printMessageToView("(The order belonges to " + rOrder.getStaffName() + ")");
         pause(2);
         return;
      }

      int orderState = rOrder.getState();
      switch(orderState)
```

```
    {
        case Order.ORDER_CLOSED:
            printMessageToView("The order is already closed.");
            pause(2);
        break;
        case Order.ORDER_CANCELED:
            printMessageToView("The order was canceled.");
            pause(2);
        break;
        default:
            editOrderItem(updateOrderID);
        break;
    }

}

private void editOrderItem(int editOrderID)
{
    boolean done = false;
    String key;
    int inputNumber = 0;

    while(!done)
    {
        cView.editOrderView();
        printMessageToView("Choose number:");
        key = cView.userInput();

        if(key.equalsIgnoreCase("Q"))   //Exit
        {
            done = true;
        }
        else
        {
            try
            {
                inputNumber = Integer.parseInt(key);
                switch(inputNumber)
                {
                    case 1: //add new item
                        addNewOrderItem(editOrderID);
                    break;
                    case 2:
                        deleteOrderItem(editOrderID);
                    break;
                    case 3: //showOrderDetail
```

```
                    cView.clearScreen();
                    cView.showOrderDetail(editOrderID);
                    printMessageToView("Please enter something to exit.");
                    key = cView.userInput();
                 break;
                 default:
                    printMessageToView("Choose 1 to 4.");
                    pause(2);
                 break;
              }
           }
           catch(Exception e)
           {
              printMessageToView("Choose 1 to 4.");
              pause(2);
           }
        }
     }
  }

  public void addNewOrderItem(int orderID)
  {
     boolean done = false;
     int addItemID = 0;
     byte addItemQuantity = 0;
     MenuItem    rNewItem = null;
     String key;

     while(!done)
     {
        cView.addOrderItemView();
        //input menu id
        while(addItemID == 0)
        {
           try
           {
              printMessageToView("Choose MenuID:");
              key = cView.userInput();
              if(key.equalsIgnoreCase("Q"))   //Exit
              {
                 printMessageToView("Transaction canceled.");
                 pause(2);
                 return;
              }
              addItemID = Integer.parseInt(key);
              rNewItem = cDatabase.findMenuItemByID(addItemID);
```

```
        if(rNewItem == null)
        {
            printErrorMessageToView("MenuID[" + addItemID + "]is not found.");
            addItemID = 0;
        }
    }
    catch(Exception e)
    {
        printErrorMessageToView("Enter valid id number.");
    }
}

//input quantity
while(addItemQuantity == 0)
{
    try
    {
        printMessageToView("Enter quantity:");
        key = cView.userInput();
        if(key.equalsIgnoreCase("Q"))   //Exit
        {
            printMessageToView("Transaction canceled.");
            pause(2);
            return;
        }
        addItemQuantity = Byte.parseByte(key);
        if( addItemQuantity <= 0)
        {
            printErrorMessageToView("Enter positive number.");
            addItemQuantity = 0;
        }
    }
    catch(NumberFormatException nfe)
    {
        printErrorMessageToView("Quantity is too large!!");
    }
    catch(Exception e)
    {
        printErrorMessageToView("Enter valid id number.");
    }
}

//Check all input
printMessageToView("MenuID:" + addItemID + " MenuName:" + rNewItem.getName()
            + " Quantity:" + addItemQuantity);
```

```
            printMessageToView("OK?(yes:y)");
            key = cView.userInput();
            if(!key.equalsIgnoreCase("Y"))
            {
               printMessageToView("canceled.");
               addItemID = 0;
               addItemQuantity = 0;
               rNewItem = null;
               continue; //back to beginning the loop
            }


            //////////ADD!!!(database)/////////////
            cDatabase.addOrderItem(orderID, rNewItem, addItemQuantity);

            printMessageToView("Add another item?(yes:y)");
            key = cView.userInput();
            if(!key.equalsIgnoreCase("Y"))//finish adding item
            {
               done = true;
            }
            else//continue adding item
            {
               //initialize
               addItemID = 0;
               addItemQuantity = 0;
               rNewItem = null;
            }
         }
      }

      public void deleteOrderItem(int orderID)
      {
         String key;
         boolean done = false;
         int deleteNo = 0;

         Order rOrder = cDatabase.findOrderByID(orderID);
         if( currentUserID != rOrder.getStaffID())
         {
            printErrorMessageToView("You are not eligible to delete the order.");
            printMessageToView("(The order belonges to " + rOrder.getStaffName() + ")");
            pause(2);
            return;
         }
```

```
   while(!done)
   {
      try
      {
         //show order detail///////
         cView.deleteOrderItemView();
         cView.showOrderDetail(orderID);
         printMessageToView("Choose number to delete or type Q to exit):");
         key = cView.userInput();
         if(key.equalsIgnoreCase("Q"))   //Exit
         {
            return;
         }
         deleteNo = Integer.parseInt(key) - 1;   //index actually starts from zero
         if(!cDatabase.deleteOrderItem(orderID, deleteNo))
         {
            printErrorMessageToView("Not found.");
            pause(2);
            continue;   //delete error
         }
         cView.deleteOrderItemView();
         cView.showOrderDetail(orderID);
         printMessageToView("Deleted.");
         printMessageToView("Delete another item?(yes:y)");
         key = cView.userInput();
         if( !key.equalsIgnoreCase("Y"))
            done = true;
      }
      catch(Exception e)
      {
         printErrorMessageToView("Enter valid integer.");
         pause(2);
      }
   }
}

private void closeOrder()
{
   cView.closeOrderView();

   int closeOrderID = findOrder();
   if(closeOrderID == -1) return;

   Order rOrder = cDatabase.findOrderByID(closeOrderID);
   if( currentUserID != rOrder.getStaffID())
   {
```

```
            printErrorMessageToView("You are not eligible to delete the order.");
            printMessageToView("(The order belonges to " + rOrder.getStaffName() + ")");
            pause(3);
            return;
        }

        if(rOrder.getState() != 0)
        {
            printMessageToView("The order is already closed or canceled.");
            pause(2);
            return;
        }

        printMessageToView("Are you sure to close this order?(YES:y)");
        String key = cView.userInput();

        if(key.equalsIgnoreCase("Y"))//back to previous menu
        {
            cDatabase.closeOrder(closeOrderID);
            printMessageToView("The order have been closed.");
            pause(2);
        }
    }

    private void cancelOrder()
    {
        cView.cancelOrderView();

        int cancelOrderID = findOrder();
        if(cancelOrderID == -1) return;

        Order rOrder = cDatabase.findOrderByID(cancelOrderID);
        if( currentUserID != rOrder.getStaffID())
        {
            printErrorMessageToView("You are not eligible to delete the order.");
            printMessageToView("(The order belonges to " + rOrder.getStaffName() + ")");
            pause(3);
            return;
        }

        if( rOrder.getState() != 0)
        {
            printMessageToView("The order is already closed or canceled.");
            pause(2);
            return;
        }
```

```
    printMessageToView("Are you sure to cancel this order?(YES:y)");
    String key = cView.userInput();

    if(key.equalsIgnoreCase("Y"))//back to previous menu
    {
        cDatabase.cancelOrder(cancelOrderID);
        printMessageToView("The order have been canceled.");
        pause(2);
    }
}

private int findOrder()
{
    String key;
    int imputID = -1;

    while(imputID == -1)
    {
        printMessageToView("Choose orderID:");
        key = cView.userInput();

        if(key.equalsIgnoreCase("Q"))//back to previous menu
        {
            break;
        }

        try
        {
            imputID = Integer.parseInt(key);
            if( imputID < 0)
            {
                printErrorMessageToView("ID must be positive integer.");
                imputID = -1;//initiarise
                continue;   //back to begining of loop
            }

            Order rOrder = cDatabase.findOrderByID(imputID);
            if(rOrder == null)//order not found
            {
                printErrorMessageToView("OrderID[" + imputID + "]is not found.");
                imputID = -1;//initiarise
                continue;   //back to begining of loop
            }
        }
        catch(Exception e)
```

```
        {
            printMessageToView("Enter valid Integer.");
        }
    }
    return imputID;
}

 /*************************************************************
 * Edit menu item mode
 *************************************************************/
//-----------------------------------------------------------
// Choose edit mode (1:Add 2:Update 3:Delete)
//-----------------------------------------------------------
private void chooseEditMenuItemMode()
{
    String  key;
    int     inputNumber = 0;

    cView.choseEditMenuView();
    printMessageToView("Choose number:");
    key = cView.userInput();

    if(key.equalsIgnoreCase("Q"))   //Back to main menu
    {
        scene = SCENE_MAIN_MENU;
        return;
    }

    while(inputNumber == 0)
    {
        try
        {
            inputNumber = Integer.parseInt(key);
            switch(inputNumber)
            {
                case 1: //add new employee
                    addNewMenuItem();
                break;
                case 2:
                    updateMenuItem();
                break;
                case 3:
                    deleteMenuItem();
                break;
                default:
                    printMessageToView("Choose 1 to 3:");
```

```
                key = cView.userInput();
                break;
            }
        }
        catch(Exception e)
        {
            printMessageToView("Choose 1 to 3:");
            key = cView.userInput();
        }
    }
}

//-----------------------------------------------------------
// Make and check new ID (used by addMenuItem method only)
//-----------------------------------------------------------
private int generateMenuID()
{
    int newID = 0;
    String key;

    printMessageToView("Choose ID for new item:");
    key = cView.userInput();

    while(newID == 0)
    {
        //Back to main menu
        if(key.equalsIgnoreCase("Q"))
            return 0;
        try
        {
            newID = Integer.parseInt(key);
            if(newID > 9999)
            {
                printMessageToView( "Please enter less than 10000");
                key = cView.userInput();
                newID = 0;
            }
            else
            {
                //Check if there is same ID
                MenuItem   rMenuItem = cDatabase.findMenuItemByID(newID);
                //if(found)
                if(rMenuItem != null)
                {
                    printMessageToView( "ID:" + newID + "is already used by " +
rMenuItem.getName());
```

```
                printMessageToView("Please try another number:");
                key = cView.userInput();
                newID = 0;
            }
        }
      }
      catch(Exception e)
      {
         printMessageToView("Please enter valid integer.");
         key = cView.userInput();
      }

   }
   return newID;
}
//-------------------------------------------------------
// Add new menu item
//-------------------------------------------------------
private void addNewMenuItem()
{
   int newID=0;
   String   newName;
   double  newPrice;
   byte     newType;
   String key;

   cView.addMenuItemView();
   //cView.clearScreen();
   //displayTi("********** Add new item **********");

   boolean done = false;
   //-------------------- loop until new item is added or enter "Q" ----------
   while(!done)
   {
      newID = generateMenuID();
      if (newID == 0)
      {
         //back to mein menu
         //scene = SCENE_MAIN_MENU;
         return;
      }

      printMessageToView("Enter item name:");
      newName = cView.userInput();

      newPrice = 0;
```

```
        printMessageToView("Enter price:");
        key = cView.userInput();
        while(newPrice == 0)
        {
           try
           {
              newPrice = Double.parseDouble(key);
              if(newPrice <= 0)
              {
                 printMessageToView("Enter positive number:");
                 key = cView.userInput();
                 newPrice = 0;
              }
           }
           catch(Exception e)
           {
              printMessageToView("Enter valid number:");
              key = cView.userInput();
           }
        }

        newType = 0;
        printMessageToView("Enter item type(1:MAIN 2:DRINK 3:ALCOHOL
4:DESSERT):");
        key = cView.userInput();
        while(newType == 0)
        {
           try
           {
              newType = Byte.parseByte(key);
              if(newType < 1 || 4< newType)
              {
                 printMessageToView("Enter 1 to 4:");
                 key = cView.userInput();
                 newType = 0;
              }
           }
           catch(Exception e)
           {
              printMessageToView("Enter valid number:");
              key = cView.userInput();
           }
        }

        //Check all input
        printMessageToView("NewID:" + newID);
```

```
        printMessageToView("New item name:" + newName);
        printMessageToView("New item price:" + newPrice);
        switch(newType)
        {
           case MenuItem.MAIN:
              printMessageToView("New item type:MAIN");
              break;
           case MenuItem.DRINK:
              printMessageToView("New item type:DRINK");
              break;
           case MenuItem.ALCOHOL:
              printMessageToView("New item type:ALCOHOL");
              break;
           case MenuItem.DESSERT:
              printMessageToView("New item type:DESSERT");
              break;
        }

        printMessageToView("\nOK? (Y:yes)");
        key = cView.userInput();

        if(key.equalsIgnoreCase("Y"))
        {
           try
           {
              cDatabase.addMenuItem(newID, newName, newPrice, newType);
              printMessageToView("New menu item is added.");
           }
           catch(DatabaseException dbe)
           {
              printErrorMessageToView("Add menu item error.");
           }

           done = true;
        }
     }
     //---------------- MenuItem is added (loop end)----------------------------

     printMessageToView("Please enter something to exit");
     key = cView.userInput();
     //scene = SCENE_MAIN_MENU;
  }


  //-----------------------------------------------------
  // Edit menu item
```

```
//---------------------------------------------------------
private void updateMenuItem()
{
   String  key = "";
   int     inputNumber = 0;
   MenuItem   rMenuItem = null;

   cView.showMenuList();
   printMessageToView("----------------------------------------");

   while(rMenuItem == null)
   {
      printMessageToView("Choose menu ID to edit:");
      key = cView.userInput();
      if(key.equalsIgnoreCase("Q"))   //Back to main menu
      {
         printMessageToView("Transaction is canceled.");
         pause(2);
         return;
      }

      try
      {
         //findUser
         inputNumber = Integer.parseInt(key);
         rMenuItem = cDatabase.findMenuItemByID(inputNumber);

         if(rMenuItem == null)
         {
            printErrorMessageToView("ID is not found.");
         }
      }
      catch(Exception e)
      {
         printErrorMessageToView("ID must be valid number.");
      }
   }


      //-------------- Choose Edit number -----------------
      cView.editMenuItemView(rMenuItem);
      /*cView.clearScreen();
      cView.showMenuItemData(rMenuItem);
      printMessageToView("\nChoose Edit number\n");
      printMessageToView("1:Name");
      printMessageToView("2:Price");
```

```
    printMessageToView("3:Type");
    printMessageToView("4:Set promotion price");
    printMessageToView("5:Reset item state");
    printMessageToView("Q:Quit");*/
    printMessageToView("Choose Edit number:");
    inputNumber = 0;

    while(inputNumber == 0)
    {
        key = cView.userInput();
        if(key.equalsIgnoreCase("Q"))   //Back to main menu
        {
            printMessageToView("Transaction is canceled.");
            pause(2);
            return;
        }

        try{
            inputNumber = Integer.parseInt(key);
            if(inputNumber < 1 || 5 < inputNumber)
            {
                inputNumber = 0;
                printMessageToView("Enter 1 to 5:");
            }
        }
        catch(Exception e)
        {
            printMessageToView("Input valid integer:");
        }
    }
    //-------------- End choosing edit number -----------------

    //-------------- Edit item start -----------------

boolean done = false;
while(!done)
{
    cView.clearScreen();
    cView.showMenuItemData(rMenuItem);
    printMessageToView("----------------------------------------");
    try
    {
        switch(inputNumber)
        {
            case 1: //edit name
                printMessageToView("Input new name:");
```

```
                key = cView.userInput();
                cDatabase.editMenuItemData(rMenuItem, cDatabase.EDIT_ITEM_NAME, key);
                cView.showMenuItemData(rMenuItem);
                printMessageToView("Please enter something to exit");
                key = cView.userInput();
            break;
            case 2: //edit price
                printMessageToView("Input new price:");
                key = cView.userInput();
                cDatabase.editMenuItemData(rMenuItem, cDatabase.EDIT_ITEM_PRICE, key);
                cView.showMenuItemData(rMenuItem);
                printMessageToView("Please enter something to exit");
                key = cView.userInput();
            break;
            case 3: //edit type
                printMessageToView("Input new type(1:Main 2:Drink 3:Alcohol 4:Dessert):");
                key = cView.userInput();
                cDatabase.editMenuItemData(rMenuItem, cDatabase.EDIT_ITEM_TYPE, key);
                cView.showMenuItemData(rMenuItem);
                printMessageToView("Please enter something to exit");
                key = cView.userInput();
            break;
            case 4:
                printMessageToView("Input promotion price( normaly $" +
rMenuItem.gerRegularPrice() + "):");
                key = cView.userInput();
                double promotionPrice = Double.parseDouble(key);
                if(promotionPrice >= rMenuItem.gerRegularPrice())
                {
                    printErrorMessageToView("Promotion Price(" + promotionPrice
                        + ") should be lower than normal price(" + rMenuItem.gerRegularPrice() +
")!!");
                  pause(2);
                  continue;
                }
                else if(promotionPrice < 0)
                {
                    printErrorMessageToView("Enter positive number.");
                  pause(2);
                  continue;
                }
                else
                {
                    /////database//////
                    cDatabase.setMenuItemAsPromotionItem(rMenuItem, promotionPrice);
                    cView.showMenuItemData(rMenuItem);
```

```
                    printMessageToView("Please enter something to exit");
                    key = cView.userInput();
                }
            break;
            case 5:
                cDatabase.resetMenuState(rMenuItem);
                cView.showMenuItemData(rMenuItem);
                printMessageToView("Item state have been initialized.");
                 printMessageToView("Please enter something to exit");
                 key = cView.userInput();
            break;
            default:
                printMessageToView("This line must not be execute!! Please check
program.(Controller class)");
                pause(2);
            break;
          }
          done = true;
        }
        catch(DatabaseException dbe)
        {
          printErrorMessageToView(dbe.getErrMessage());
          pause(2);
        }
        catch( Exception e)
        {
          printErrorMessageToView("" + key +  "" + "is not acceptable. Please enter only
number.");
          pause(2);
        }
        //-------------- Edit item end -----------------
      }
      //back to main menu
      //scene = SCENE_MAIN_MENU;
   }

   //-----------------------------------------------------------
   // Delete menuItem
   //-----------------------------------------------------------
   private void deleteMenuItem()
   {
      String  key;
      int     inputNumber = 0;
      MenuItem   rMenuItem = null;

       while(inputNumber == 0)
```

```
{
   try
   {
      //findUser
      while(rMenuItem == null)
      {
         cView.showMenuList();
         printMessageToView("Choose menu item ID to delete:");
         key = cView.userInput();
         if(key.equalsIgnoreCase("Q"))   //Back to main menu
         {
            //scene = SCENE_MAIN_MENU;
            return;
         }
         inputNumber = Integer.parseInt(key);
         rMenuItem = cDatabase.findMenuItemByID(inputNumber);
         if(rMenuItem == null)
         {
            printMessageToView("Item is not found.:");
            pause(2);
         }
      }

      printMessageToView("MenuItem ID:" + rMenuItem.getID());
      printMessageToView(" Name:" + rMenuItem.getName());
      printMessageToView("Price:" + rMenuItem.getPrice());
      printMessageToView("will be deleted. OK? (YES:y)");

      key = cView.userInput();
      if(!key.equalsIgnoreCase("Y"))
      {
         printMessageToView("The transaction is canceled.");
         pause(2);
         //scene = SCENE_MAIN_MENU;
         return;
      }

      cDatabase.deleteMenuItem(rMenuItem);

      printMessageToView("Deleted.");
      pause(2);
   }
   catch(Exception e)
   {
      printMessageToView("ID must be valid number.");
      pause(2);
```

```
        }
    }
}

 /**************************************************************
 * Display database data
 **************************************************************/
private void showMenuList()
{
    cView.showMenuList();
    printMessageToView("Please enter something to exit.");
    cView.userInput();
    // back to main menu
    scene = SCENE_MAIN_MENU;
}

private void showStaffList()
{
    cView.showStaffList();
    printMessageToView("Please enter something to exit.");
    cView.userInput();
    // back to main menu
    scene = SCENE_MAIN_MENU;
}


private void printMessageToView(String message)
{
    cView.displayMessage(message);
}

private void printErrorMessageToView(String message)
{
    cView.displayErrorMessage(message);
}

// create pause for some seconds
private void pause( long secs)
{
    try
    {
        Thread.currentThread().sleep(secs * 1000);
    }
    catch(InterruptedException e)
    {
      e.printStackTrace();
```

```
    }
}
/*************************************************************
 * Generate reports
 *************************************************************/
private void generateReports()
{
    String key;
    int selection = 0;

    cView.generateReportView();
    printMessageToView("Choose number:");
    while(selection == 0)
    {
        try
        {
            key = cView.userInput();
            if(key.equalsIgnoreCase("Q"))
            {
                //Back to main menu
                scene = SCENE_MAIN_MENU;
                return;
            }

            selection = Integer.parseInt(key);

            String filename;

            switch(selection)
            {
                case 1:
                    generateSalesReports();

                break;
                case 2:
                    generatePaymentReports();
                break;
                default:
                    selection = 0;
                    printMessageToView("Choose 1 or 2:");
                break;
            }
        }
        catch(Exception e)
        {
            printMessageToView("Choose 1 or 2:");
```

```
        }
    }
}

private void generateSalesReports()
{
    String key;

    cView.showOrderList();
    printMessageToView("Print out? (YES:y)");
    key = cView.userInput();

    if(key.equalsIgnoreCase("Y"))
    {
        if(!cDatabase.checkIfAllOrderClosed())
        {
            printMessageToView("All orders must be closed or canceled before generate report.");
            printMessageToView("Do you want to close all orders? (YES:y)");
            key = cView.userInput();
            if(key.equalsIgnoreCase("Y"))
            {
                cDatabase.closeAllOrder();
            }
            else
            {
                scene = SCENE_MAIN_MENU;
                return;
            }
        }

        try
        {
            String filename = cDatabase.generateOrderReport(todaysDate);
            printMessageToView("File <" + filename + "> has been generated.");
            printMessageToView("Done.");
            printMessageToView("Please enter something to exit.");
            key = cView.userInput();
        }
        catch(DatabaseException de)
        {
            printErrorMessageToView(de.getErrMessage());
            pause(3);
        }
    }
    // back to main menu
    scene = SCENE_MAIN_MENU;
```

```
    }

    private void generatePaymentReports()
    {
        String key;

        cView.showPaymentList();
        printMessageToView("Print out? (YES:y)");
        key = cView.userInput();

        if(key.equalsIgnoreCase("Y"))
        {
            //if(!cDatabase.checkIfAllOrderClosed())
            if(!cDatabase.checkIfAllStaffCheckout())
            {
                printMessageToView("There still exist some staff being active.");
                printMessageToView("All staff must be checked out before generate a payment
report.");
                printMessageToView("Do you want to make all staff finished work? (YES:y)");
                key = cView.userInput();
                if(key.equalsIgnoreCase("Y"))
                {
                    cDatabase.forthClockOutAllStaff();

                }
                else
                {
                    scene = SCENE_MAIN_MENU;
                    return;
                }
            }

            try
            {
                String filename = cDatabase.generatePaymentReport(todaysDate);
                printMessageToView("File <" + filename + "> has been generated.");
                printMessageToView("Please enter something to exit.");
                key = cView.userInput();
            }
            catch(DatabaseException de)
            {
                printErrorMessageToView(de.getErrMessage());
                pause(3);
            }
        }
        // back to main menu
```

```
      scene = SCENE_MAIN_MENU;
   }
}
```

DATABASE:

```
/**
 * Kazunori Hayashi
* Version 1.0 29/7/2013
 */
import java.util.*;
import java.io.*;
import java.lang.*;
import java.util.Comparator;

public class Database
{
   private final static String STAFF_FILE = "dataFiles/staff.txt";
   private final static String MANAGER_FILE = "dataFiles/manager.txt";
   private final static String MENU_FILE = "dataFiles/menu_item.txt";
   private final static String REPORT_FILE = "dataFiles/reports/report_";
   private final static String PAYMENT_FILE = "dataFiles/reports/payment_";
   private final static String WAGE_INFO_FILE = "dataFiles/wage_info.txt";

   private ArrayList<Staff> staffList = new ArrayList<Staff>();
   private ArrayList<MenuItem> menuList = new ArrayList<MenuItem>();
   private ArrayList<Order> orderList = new ArrayList<Order>();

   private Date    date;
   int     todaysOrderCounts;

/*************************************************************************
    * Constructor

*************************************************************************/
   public Database()
   {
      date = new Date();
      todaysOrderCounts = 0;  //Load order file??
   }

/*************************************************************************
    * Getter

*************************************************************************/
    public ArrayList<Staff> getStaffList()
```

```
 {
    return staffList;
 }

 public ArrayList<MenuItem> getMenuList()
 {
    return menuList;
 }

 public ArrayList<Order> getOrderList()
 {
    return orderList;
 }

 public int getTodaysOrderCount()
 {
    return this.todaysOrderCounts;
 }

//-------------------------------------------------------
// Find staff from ID
//-------------------------------------------------------
public Staff   findStaffByID(int id)
{
   Iterator<Staff> it = staffList.iterator();
   Staff          re = null;
   boolean        found = false;

   if(id < 0){
      return null;
   }

   while (it.hasNext() && !found) {
      re = (Staff)it.next();
      if( re.getID() == id)
      {
         found = true;
      }
   }

   if(found)
      return re;
   else
      return null;
}
```

```
//---------------------------------------------------------
// Find menu item from ID
//---------------------------------------------------------
public MenuItem   findMenuItemByID(int id)
{
   Iterator<MenuItem> it = menuList.iterator();
   MenuItem          re = null;
   boolean         found = false;

   if(id < 0){
      return null;
   }

   while (it.hasNext() && !found) {
      re = (MenuItem)it.next();
      if( re.getID() == id)
      {
         found = true;
      }
   }

   if(found)
      return re;
   else
      return null;
}

//---------------------------------------------------------
// Find order from ID
//---------------------------------------------------------
public Order   findOrderByID(int id)
{
   Iterator<Order> it = orderList.iterator();
   Order          re = null;
   boolean         found = false;

   if(id < 0){
      return null;
   }

   while (it.hasNext() && !found) {
      re = it.next();
      if( re.getOrderID() == id)
      {
         found = true;
      }
```

```
        }

        if(found)
            return re;
        else
            return null;
    }

/*****************************************************************************
    * Manipurate datas

*****************************************************************************/
    //-------------------------------------------------------------
    // Staff information
    //-------------------------------------------------------------
    //edit staff data
    // rStaff reference the staff
    // which 1:Lastname 2:Firstname 3:passward
    public final static int EDIT_LAST_NAME = 1;
    public final static int EDIT_FIRST_NAME = 2;
    public final static int EDIT_PASSWORD = 3;

    public void editStaffData(int staffID, String newPassword, String newFirstName, String
newLastName) throws DatabaseException
    {
        Staff  rStaff = findStaffByID(staffID);
        rStaff.setPassword(newPassword);
        rStaff.setLastName(newLastName);
        rStaff.setFirstName(newFirstName);

        try
        {
            if(rStaff instanceof Manager)
            //if(rStaff.getClass().getName().equalsIgnoreCase("Manager"))
            updateStaffFile(true);//update manager file
            else
            updateStaffFile(false);//update employee file
        }
        catch(DatabaseException dbe)
        {
            throw dbe;
        }
    }

    public void editStaffData(Staff rStaff, int which, String newData) throws DatabaseException
    {
```

```
    switch(which)
    {
       case EDIT_LAST_NAME:
         rStaff.setLastName(newData);
       break;
       case EDIT_FIRST_NAME:
         rStaff.setFirstName(newData);
       break;
       case EDIT_PASSWORD:
         rStaff.setPassword(newData);
       break;
       default:
       break;
    }

    try
    {
       if(rStaff instanceof Manager)
       //if(rStaff.getClass().getName().equalsIgnoreCase("Manager"))
         updateStaffFile(true);//update manager file
       else
         updateStaffFile(false);//update employee file
    }
    catch(DatabaseException dbe)
    {
       throw dbe;
    }
}

public void deleteStaff(Staff rStaff) throws DatabaseException
{
    boolean isManager = false;
    staffList.remove(rStaff);
    //if(rStaff.getClass().getName().equalsIgnoreCase("Manager"))
   if(rStaff instanceof Manager)
    isManager = true;
   try
   {
      updateStaffFile(isManager);
   }
   catch(DatabaseException dbe)
   {
      throw dbe;
   }
}
```

```
    public void addStaff(int newID, String newPassward, String newFirstName, String
newLastName, boolean isManager) throws DatabaseException
    {
       Staff newStaff;
       if(isManager)
          newStaff = new Manager(newID, newLastName, newFirstName, newPassward);
       else
          newStaff = new Employee(newID, newLastName, newFirstName, newPassward);
       staffList.add(newStaff);
       if(newStaff instanceof Manager)
       //if(newStaff.getClass().getName().equalsIgnoreCase("Manager"))
          isManager = true;
      try
       {
          updateStaffFile(isManager);
       }
       catch(DatabaseException dbe)
       {
          throw dbe;
       }
    }

    //-----------------------------------------------------------------
    // MenuItem
    //-----------------------------------------------------------------
    //edit menu item data
    // rMenuItem reference the MenuItem
    // which 1:name 2:price 3:type
    public final static int EDIT_ITEM_NAME = 1;
    public final static int EDIT_ITEM_PRICE = 2;
    public final static int EDIT_ITEM_TYPE = 3;

    public void editMenuItemData(int id, String newName, double newPrice, byte menuType)
throws DatabaseException
    {
       MenuItem rMenuItem = findMenuItemByID(id);
       rMenuItem.setName(newName);
       rMenuItem.setPrice(newPrice);
       rMenuItem.setType(menuType);

       /*try
       {
          updateMenuFile();
       }
       catch(DatabaseException dbe)
```

```
    {
        throw dbe;
    }*/
    }

    public void editMenuItemData(MenuItem rMenuItem, int which, String newData) throws
DatabaseException
    {
        try
        {
            switch(which)
            {
                case EDIT_ITEM_NAME:
                    rMenuItem.setName(newData);
                    break;
                case EDIT_ITEM_PRICE:
                    double newPrice = Double.parseDouble(newData);
                    if(newPrice < 0)
                        throw new DatabaseException("Price must be positive number");
                    else
                        rMenuItem.setPrice(newPrice);
                    break;
                case EDIT_ITEM_TYPE:
                    byte newType = Byte.parseByte(newData);
                    if(newType < MenuItem.MAIN || MenuItem.DESSERT < newType)
                        throw new DatabaseException("Type must be between " + MenuItem.MAIN
                                + " and " + MenuItem.DESSERT + ")");
                    else
                        rMenuItem.setType(Byte.parseByte(newData));
                    break;
                default:
                    break;
            }
        }
        catch(DatabaseException e)
        {
            throw e;
        }
        catch(Exception e)
        {
            throw new DatabaseException(e.getMessage());
        }
    }

    public void setMenuItemAsPromotionItem(MenuItem rMenuItem, double price)
    {
```

```
        rMenuItem.setState(MenuItem.PROMOTION_ITEM, price);
    }

    public void resetMenuState(MenuItem rMenuItem)
    {
        rMenuItem.resetState();
    }

    public void deleteMenuItem(MenuItem rMenuItem) throws DatabaseException
    {
        menuList.remove(rMenuItem);
        /*try
        {
            updateMenuFile();
        }
        catch(DatabaseException dbe)
        {
            throw dbe;
        }*/
    }

    public void addMenuItem(int newID, String newName, double newPrice, byte newType)
throws DatabaseException
    {
        MenuItem newMenuItem = new MenuItem(newID, newName,newPrice, newType);
        menuList.add(newMenuItem);
        Collections.sort(menuList, new MenuItemComparator());
        /*try
        {
            updateMenuFile();
        }
        catch(DatabaseException dbe)
        {
            throw dbe;
        }*/
    }
    //-------------------------------------------------------------
    // Order
    //-------------------------------------------------------------
    public int addOrder(int staffID, String staffName)
    {
        int newOrderID = ++todaysOrderCounts;
        Order newOrder = new Order(staffID, staffName);
        newOrder.setOrderID( newOrderID);
        orderList.add(newOrder);
        return newOrderID;
```

```
}

public void addOrderItem(int orderID, MenuItem rItem, byte quantity)
{
   Order rOrder = findOrderByID(orderID);
   rOrder.addItem(rItem, quantity);
}

public boolean deleteOrderItem(int orderID, int index)
{
    Order rOrder = findOrderByID(orderID);
    if(rOrder == null)
      return false;
    return rOrder.deleteItem(index);
}


//Cancel order: order data is not deleted from the database(Just put cancel flag on)
public boolean cancelOrder(int orderID)
{
   Order rOrder = findOrderByID(orderID);
  if(rOrder == null)
     return false;
   rOrder.setState(Order.ORDER_CANCELED);
   return true;
}
//Delete order: order data is deleted from the database
public boolean deleteOrder(int orderID)
{
   Order rOrder = findOrderByID(orderID);
  if(rOrder == null)
     return false;
   orderList.remove(rOrder);
   todaysOrderCounts--;
   return true;
}

public boolean closeOrder(int orderID)
{
   Order rOrder = findOrderByID(orderID);
  if(rOrder == null)
     return false;
   rOrder.setState(Order.ORDER_CLOSED);
   return true;
}
```

```
public void closeAllOrder()
{
  Iterator<Order> it = orderList.iterator();
  Order          re = null;

  while (it.hasNext()) {
     re = it.next();
     if( re.getState() == 0)//neither closed and canceled
     {
        re.setState(Order.ORDER_CLOSED);
     }
  }
}

public int getOrderState(int orderID)
{
   Order  re = findOrderByID(orderID);
   if(re == null)
      return -1;
   return re.getState();
}

public double getOrderTotalCharge(int orderID)
{
   Order  re = findOrderByID(orderID);
   if(re == null)
      return -1;
   return re.getTotal();
}

public boolean checkIfAllOrderClosed()
{
  Iterator<Order> it = orderList.iterator();
  Order          re = null;

  while (it.hasNext()) {
     re = it.next();
     if( re.getState() == 0)//neither closed and canceled
     {
        return false;
     }
  }
  return true;
}

public boolean checkIfAllStaffCheckout()
```

```
    {
      Iterator<Staff> it = staffList.iterator();
      Staff          re = null;

      while (it.hasNext()) {
         re = it.next();
         if( re.getWorkState() == Staff.WORKSTATE_ACTIVE)
         {
            return false;
         }
      }
      return true;
    }

    public void forthClockOutAllStaff()
    {
       Iterator<Staff> it = staffList.iterator();
       Staff          re = null;

       while (it.hasNext()) {
         re = it.next();
         if( re.getWorkState() == Staff.WORKSTATE_ACTIVE)
         {
            re.clockOut();
         }
       }
    }

/*****************************************************************************
   * File load
   *****************************************************************************/
   public void loadFiles() throws DatabaseException
   {
      loadStaffFile();
      loadManagerFile();
      Collections.sort(staffList, new StaffComparator());
      loadMenuFile();
      loadWageInfoFile();
   }

   private void loadStaffFile() throws DatabaseException
   {
         try {
         BufferedReader reader = new BufferedReader(new FileReader(STAFF_FILE));
         String line = reader.readLine();
```

```
            while (line != null) {
                String[] record = line.split(",");

                String id = record[0].trim();
                String passward = record[1].trim();
                String firstName = record[2].trim();
                String lastName = record[3].trim();

                // Add the data from file to the registerCourses array list
                Employee rEmployee = new Employee(Integer.parseInt(id),lastName, firstName,
passward);
                staffList.add(rEmployee);
                line = reader.readLine();
            }
            reader.close();
        } catch (IOException ioe) {
            String message = ioe.getMessage() + ioe.getStackTrace();
            throw new DatabaseException(message);
        }
    }

    private void loadManagerFile() throws DatabaseException
    {
        try {
            BufferedReader reader = new BufferedReader(new FileReader(MANAGER_FILE));
            String line = reader.readLine();

            while (line != null) {
                String[] record = line.split(",");

                String id = record[0].trim();
                String passward = record[1].trim();
                String firstName = record[2].trim();
                String lastName = record[3].trim();

                // Add the data from file to the registerCourses array list
                Manager rManager = new Manager(Integer.parseInt(id),lastName,firstName,
passward);
                staffList.add(rManager);
                line = reader.readLine();
            }
            reader.close();
        } catch (IOException ioe) {
            String message = ioe.getMessage() + ioe.getStackTrace();
            throw new DatabaseException(message);
        }
```

```
    }

    private void loadMenuFile() throws DatabaseException
    {
        try {
            BufferedReader reader = new BufferedReader(new FileReader(MENU_FILE));
            String line = reader.readLine();

            while (line != null) {
                String[] record = line.split(",");

                String id = record[0].trim();
                String name = record[1].trim();
                String price = record[2].trim();
                String type = record[3].trim();

                // Add the data from file to the registerCourses array list
                MenuItem rMenuItem = new MenuItem(Integer.parseInt(id), name,
Double.parseDouble(price), Byte.parseByte(type));
                menuList.add(rMenuItem);
                line = reader.readLine();
            }
            reader.close();
        } catch (IOException ioe) {
            String message = ioe.getMessage() + ioe.getStackTrace();
            throw new DatabaseException(message);
        }
    }

    private void loadWageInfoFile() throws DatabaseException
    {
        try {
            BufferedReader reader = new BufferedReader(new FileReader(WAGE_INFO_FILE));
            String line = reader.readLine();

            while (line != null) {
                String[] record = line.split(",");

                String id = record[0].trim();
                String rate = record[1].trim();

                double dRate = Double.parseDouble(rate);
                int iId = Integer.parseInt(id);

                Staff rStaff = findStaffByID(iId);
                if(rStaff == null)
```

```
            {
               throw new DatabaseException("Load wage file error\n Staff ID:" + iId + " is not
found.");
            }
            rStaff.setWageRate(dRate);

            line = reader.readLine();
         }
         reader.close();
      } catch (IOException ioe) {
         String message = ioe.getMessage() + ioe.getStackTrace();
         throw new DatabaseException(message);
      }
      catch(Exception e)
      {
         String message = e.getMessage() + e.getStackTrace();
         throw new DatabaseException(message);
      }
   }


/**************************************************************************
   * File Edit
   **************************************************************************/
   public void updateStaffFile(boolean isManager) throws DatabaseException
   {
      Writer        writer;
      String        id;
      String        line;
      String        fileName;
      String        tempFileName = "dataFiles/temp.txt";

      if(isManager)
         fileName =MANAGER_FILE;
      else
         fileName = STAFF_FILE;

      Collections.sort(staffList, new StaffComparator());
      File tempFile = new File(tempFileName);

      try{
         writer = new BufferedWriter(new FileWriter(tempFile));
         Iterator it = staffList.iterator();

         while (it.hasNext())
         {
```

```
            Staff re = (Staff)it.next();

            //-------- skip writing data ----------
            if(isManager)
            {
               //skip employee data
               if(re instanceof Employee)
               //if(re.getClass().getName().equalsIgnoreCase("Employee"))
               continue;
            }
            else
            {
               //skip managere data
               //if(re.getClass().getName().equalsIgnoreCase("Manager"))
               if(re instanceof Manager)
               continue;
            }

            writer.write(re.getID() + "," + re.getPassword() + "," + re.getFirstName() + "," +
re.getLastName()+ "\r\n");
         }
         writer.flush();
         writer.close();
      }
      catch(IOException e)
      {
         String message = e.getMessage() + e.getStackTrace();
         throw new DatabaseException(message);
      }

      //delete current file
      File deleteFile = new File(fileName);
      deleteFile.delete();

      // renames temporaly file to new file
      File newFile = new File(fileName);
      tempFile.renameTo(newFile);

      updateWageFile();
   }

   public void updateWageFile() throws DatabaseException
   {
      Writer        writer;
      String        id;
      String        line;
```

```
    String        fileName;
    String        tempFileName = "dataFiles/temp.txt";

    File tempFile = new File(tempFileName);

    try{
       writer = new BufferedWriter(new FileWriter(tempFile));
       Iterator it = staffList.iterator();

       while (it.hasNext())
       {
          Staff re = (Staff)it.next();
          writer.write(re.getID() + "," + re.getWageRate() + "\r\n");
       }
       writer.flush();
       writer.close();
    }
    catch(IOException e)
    {
       String message = e.getMessage() + e.getStackTrace();
       throw new DatabaseException(message);
    }

    //delete current file
    File deleteFile = new File(WAGE_INFO_FILE);
    deleteFile.delete();

    // renames temporaly file to new file
    File newFile = new File(WAGE_INFO_FILE);
    tempFile.renameTo(newFile);
}

public void updateMenuFile() throws DatabaseException
{
    Writer        writer;
    String        id;
    String        line;
    String        tempFileName = "dataFiles/temp.txt";

    //Collections.sort(menuList, new MenuItemComparator());
    File tempFile = new File(tempFileName);

    try{
       writer = new BufferedWriter(new FileWriter(tempFile));
       Iterator it = menuList.iterator();
```

```
            while (it.hasNext())
            {
               MenuItem re = (MenuItem)it.next();

               writer.write(re.getID() + "," + re.getName() + "," + re.getPrice() + "," + re.getType()+
"\r\n");
            }
            writer.flush();
            writer.close();
         }
         catch(IOException e)
         {
            String message = e.getMessage() + e.getStackTrace();
            throw new DatabaseException(message);
         }

         //delete current file
         File deleteFile = new File(MENU_FILE);
         deleteFile.delete();

         // renames temporaly file to new file
         File newFile = new File(MENU_FILE);
         tempFile.renameTo(newFile);
      }

      public String generateOrderReport( String todaysDate) throws DatabaseException
      {
         Writer         writer = null;
         String         line;
         int            state;
         double         totalAllOrder = 0;
         String         generateFileName;
         File           newFile;
         int            orderCnt = 0;
         int            cancelCnt = 0;
         double         cancelTotal = 0;

         String[] record = todaysDate.split("/");
         String today = record[0].trim() + "_" + record[1].trim() + "_" + record[2].trim();
         generateFileName = REPORT_FILE + today + ".txt";
         newFile = new File(generateFileName);

         try{
            writer = new BufferedWriter(new FileWriter(newFile));

            line = "********** Order List (" + today + ") **********\r\n";
```

```
            writer.write(line);

            Iterator<Order> it = orderList.iterator();
            while (it.hasNext())
            {
               Order re = it.next();
               state = re.getState();
               String stateString = "";
               double totalOfEachOrder = re.getTotal();
               switch(state)
               {
                  case Order.ORDER_CLOSED:
                     stateString = "";
                     totalAllOrder += totalOfEachOrder;
                     orderCnt++;
                  break;
                  case Order.ORDER_CANCELED:
                     stateString = "Canceled";
                     cancelTotal += totalOfEachOrder;
                     cancelCnt++;
                  break;
                  default:
                     stateString = "";
                     totalAllOrder += totalOfEachOrder;
                     orderCnt++;
                  break;
               }
               String output = String.format("Order ID:%4d  StaffName:%-30s  Total:$%-5.2f
%s\r\n",
                                    re.getOrderID(),re.getStaffName(),totalOfEachOrder, stateString);
               writer.write(output);


            }
            writer.write("-----------------------------------------------------\r\n");

            writer.write("Total sales:$" + totalAllOrder + "(" + orderCnt + ")" +
                  "  Canceled:$" + cancelTotal + "(" + cancelCnt + ")\r\n");
            writer.flush();
            writer.close();
         }
         catch(IOException e)
         {
            String message = e.getMessage() + e.getStackTrace();
            newFile.delete();
            throw new DatabaseException(message);
```

```
    }
    return generateFileName;
    //System.out.println("File <" + generateFileName + "> has been generated.");
  }

  public String generatePaymentReport( String todaysDate) throws DatabaseException
  {
    Writer        writer = null;
    String        line;
    double        totalPayment = 0;
    String        generateFileName;
    File          newFile;
    int           staffNum = 0;

    String[] record = todaysDate.split("/");
    String today = record[0].trim() + "_" + record[1].trim() + "_" + record[2].trim();
    generateFileName = PAYMENT_FILE + today + ".txt";
    newFile = new File(generateFileName);

    try{
      writer = new BufferedWriter(new FileWriter(newFile));

      line = "********** Payment List (" + today + ") **********\r\n";
      writer.write(line);

      Iterator<Staff> it = staffList.iterator();
      while (it.hasNext())
      {
        Staff re = it.next();

        if(re.getWorkState() == Staff.WORKSTATE_FINISH)
        {
          double pay = re.culculateWages();
          String output = String.format("Order ID:%4d  StaffName:%-30s  Work time:%-5.2f
Pay:%-5.2f\r\n",
                                re.getID(),re.getFullName(),re.culculateWorkTime(), pay);
          writer.write(output);
          staffNum++;
          totalPayment += pay;
        }
      }
      writer.write("---------------------------------------------------\r\n");

      writer.write("Total payment:$" + totalPayment + "(" + staffNum + ")\r\n");
      writer.flush();
      writer.close();
```

```
        }
        catch(IOException e)
        {
            String message = e.getMessage() + e.getStackTrace();
            newFile.delete();
            throw new DatabaseException(message);
        }
        return generateFileName;
    }


/*****************************************************************************
    * Comparator
    *****************************************************************************/
    private class StaffComparator implements Comparator<Staff> {

        @Override
        public int compare(Staff s1, Staff s2) {
            return s1.getID() < s2.getID() ? -1 : 1;
        }
    }

    private class MenuItemComparator implements Comparator<MenuItem> {

        @Override
        public int compare(MenuItem m1, MenuItem m2) {
            return m1.getID() < m2.getID() ? -1 : 1;
        }
    }
}
```

CONCLUSION:

The main feature of our project is that it is time saving, as there is a lot of rush in restaurants and we usually have to wait for the waiters to come and take our order. By this we can directly place the order from our mobile phones and hence save a lot of time. Also, the software used in this project, python is a flexible language which can run in any type of system. It is user friendly and very easy to use. We are further going to implement this program in android platform so that this program can be installed in mobile phones. Majority of people use phones so this would be beneficial.