PARSHVANATH CHARITABLE TRUST'S

A. P. Shah Institute of Technology

Thane, 400615

Academic Year: 2022-23

Department of Computer Engineering

CSL304 SKILL BASED LAB COURSE: OBJECT ORIENTED PROGRAMMING WITH JAVA

Mini Project Report

➢ **Title of Project**	:	APM CAFE
➢ **Year and Semester**	:	S.E. (Sem III)
Group Members Name and Roll No.	:	V.ANUSHA (75), PARTH JAIN (52), MIT JAIN (51)

Table of Contents

Problem Definition – Half a page

Introduction – At least 1 page

Description – At least 2 page

Implementation details with screen-shots (stepwise) – 3 pages minimum

Learning Outcome – Half a page

<u>FOLLOW THE GUIDELINES GIVEN BELOW FOR PREPARING YOUR REPORT IN THE GIVEN FORMAT</u>

1. The project report should be neatly typed.
2. Avoid using Abbreviations.
3. The text should be justified and typed in the Font style ‘Times New Roman’ and Font size ‘12’.
4. Heading and subheading should be bold with font 14.

PROBLEM DEFINITION:

To create an awt that will help café owner to manage\ place order in an methodical way based on inputs provided by users. For this project, assume that a system is set up with a login page which will first ask you for the username and password and if you are visiting for the first time then you will see an option for signing up which will after completion of details filling direct user to the next layout where user can place order. There are various options for an user making the page and user friendly system. User can log out, place an order, view their bill as well as manage the categories of the cuisine, user can select\edit the order as per their convenience as well as exit the page after the action. Making the whole Café managing operation an undemanding process.

INTRODUCTION:

Cafe Management System is based on a concept to maintain orders and management of a particular items. This project is developed using java language and php database used. The role of the User is to maintain information including operations like modifying, deleting, updating the items records and customer order records in the system.Cafe Management System is a windows form application developed in java programming language to carry out and manage basic cafe operations efficiently. This win application is perfect for a cafe or small coffee shop owners where they are in need of an application to run and simply ease out their day to day managerial task.

Purpose of this project is to create a management system for the cafe around the world to manage the things in the cafe . It supports up to 8 bit binary input that will get from the key pad and result the data on the screen .

DESCRIPTION:

The APM CAFE is one of the updated program which is created with the all possible features and actions to perform according to user needs. While we were planning to create such a program that will the people in the coming time or in the future, the first thing that came to my mind was to make it user friendly and also get the best management system using our services that will be providing .

The world is a global village today and people are moving from one part of the world to another part of the world for business, travel, education and migration. One country’s culture is different from another country. In this situation, we have to get used to the culture of the other country , spot on while having the change . And if the its related to food and cafe , restaurant, etc .That s why to manage the food related cafe or any kind of the cafe there is always the problem of managing the stuff. That’s why we include the program and coded them efficiently so that one can create his own customized database and manage the cafe easily and solve is problem in no time . we are creating a java program for the every particular cafe that is present in the different country , it can be anything vegan cafe , non vegan cafe ,shakes cafe , cake cafe etc , our customer can customize the things accordingly . The CAFE MANAGEMENT can also see the person who is signed up and   the person is logging out . What amount of bill id being created and what all things are being added in the bill can be displayed in our program .Which order is going to which counter can be seen in our program . In what time time the customer will receive their order is being displayed in out program. Our program will be displaying these things as follows :

1 :our login page will be seen by the customer and the customer an login by putting the necessary details of the customer

2: or the customer can sign in the page

3: and now after that he can   see many details like place order etc

Our program will be handled by the cafe manager and used by the customer   for various things like, placing order and , deleting the order , viewing the order etc.

IMPLEMENTATION DETAILS

The Model

APM CAFE plays the role of model and is very simple:

APM CAFE.JAVA

The View

ConverterGUI plays the role of view. It extends JPanel and contains two text fields and a title. It also contains references to the model and controller.

ConverterGUI.java// CONVERTER GUI .JAVA

One point of interest in the ConverterGUI is the reusable display method. This method creates a top-level or desktop JFrame to hold this JPanel, then sets visibility to true. This launches a parallel thread

```
  public void display() {

      JFrame frame = new JFrame();

      frame.setTitle(title);

      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      frame.add(this);

      frame.pack();

      frame.setVisible(true);

  }
```

Here's main. When display is called a parallel thread is launched containing the GUI's listener loop. When the top-level window is closed, then main terminates.

```
    public static void main(String[] args) {
    ConverterGUI gui = new ConverterGUI();
    gui.display();
  }
```

The Controller

The ConverterController plays the role of controller. It implements the ActionListener interface. This means two things. First, the controller can register itself with controls that it is interested in:

```
  celsiusField.addActionListener(controller);
  farField.addActionListener(controller);
```

Second, it must implement the actionPerformed method:

```
public void actionPerformed(ActionEvent e) {
  // interact with model and view here
}
```

Each time the user types something into one of the text fields then presses the "enter" key, the Java VM creates an instance of ActionEvent containing the identity of the control that fired the event, then passes this object to the actionPerformed method of every registered listener.

Here's the complete implementation:

ConverterController.java

An Improvement

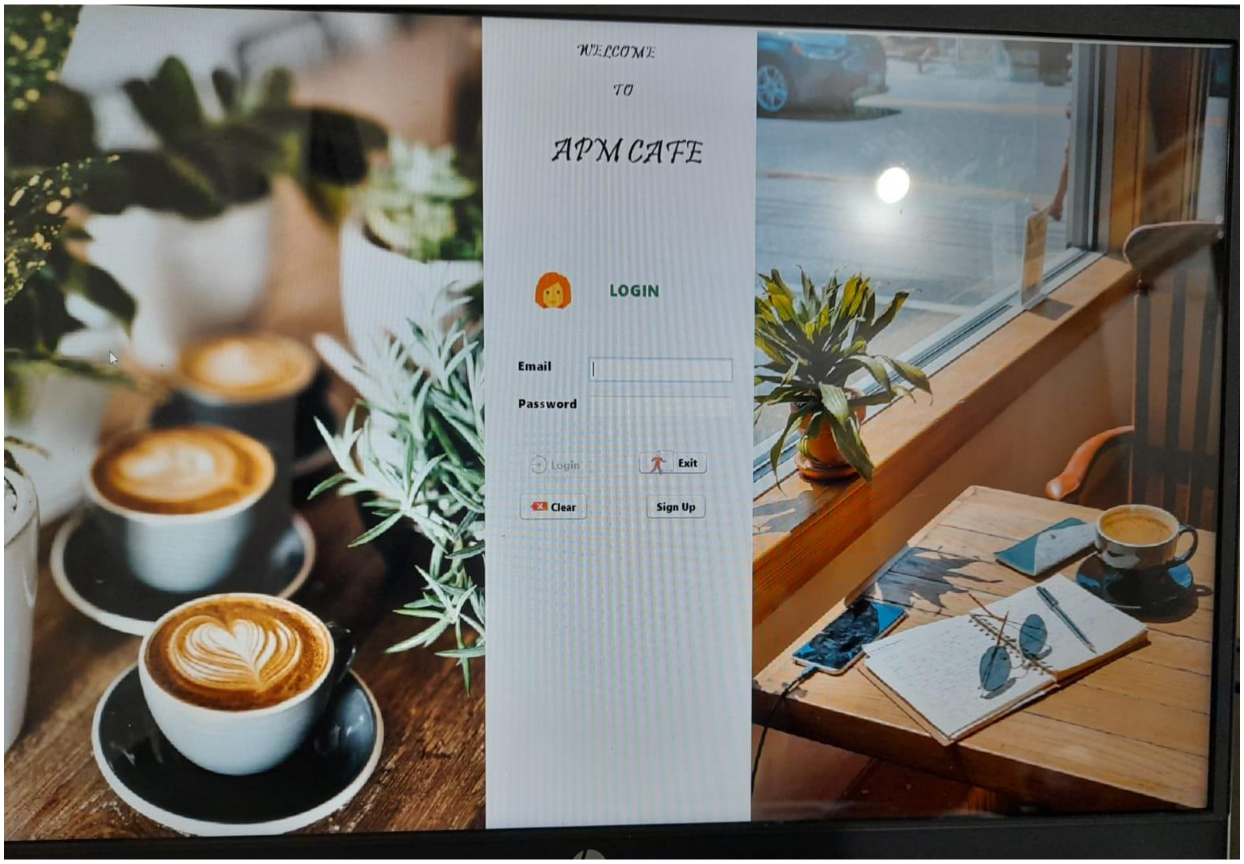
We can improve the design of a view by declaring the controllers to be inner classes.

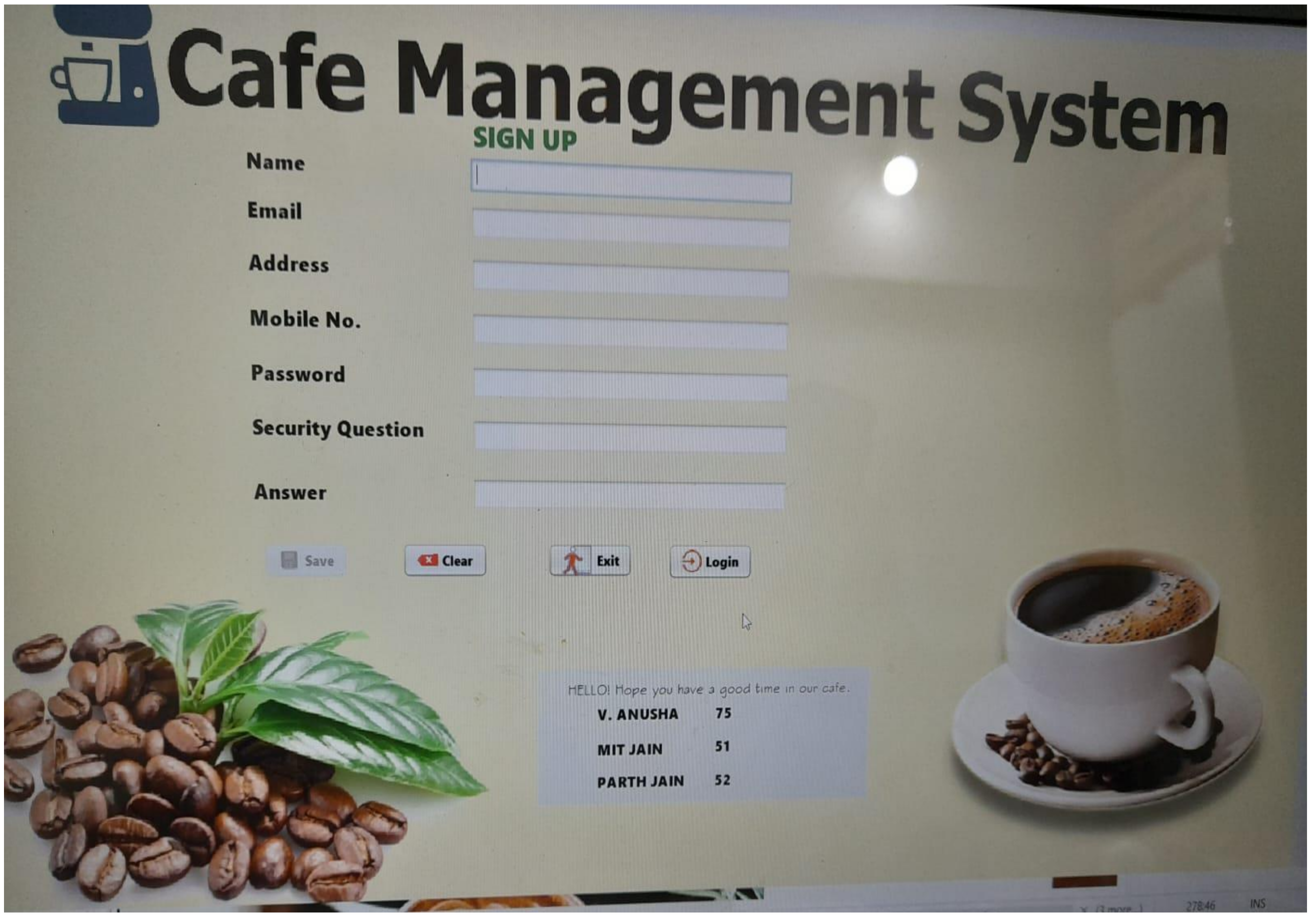
Instances of inner classes (called products) are associated with instances of the outer class that created them (called factories). Products have full access to the fields of their associated factories, private or otherwise. It is not necessary for an "inner" controller to constantly call getter and setter methods to access the controls and models of its outer view class.

Here's an example:

ConverterGUI.java (version 2)

Adding Menus





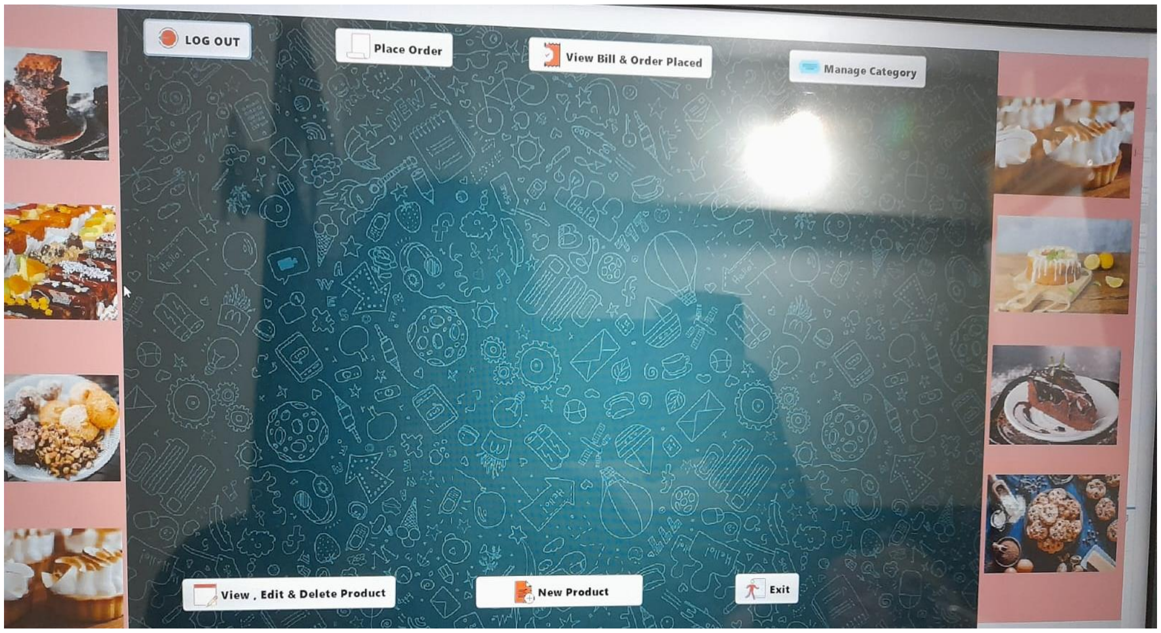
Cafe Management System

SIGN UP

Name	
Email	
Address	
Mobile No.	
Password	
Security Question	
Answer	

HELLO! Hope you have a good time in our cafe.

V. ANUSHA	75
MIT JAIN	51
PARTH JAIN	52



Here's the implementation:

[ConverterGUI.java]

The menu bar is a field:

private JMenuBar menuBar;

Note that the display method attaches the menu bar to the desktop frame:

```
public void display() {

    JFrame frame = new JFrame();

    frame.setTitle(title);

    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    frame.setJMenuBar(menuBar);

    frame.add(this);

    frame.pack();
```

```
      frame.setVisible(true);

   }
```

The menu bar is initialized in the constructor with a call to makeMenus:

```
   private void makeMenus() {

      menuBar = new JMenuBar();

      JMenu cmmdMenu = new JMenu("Commands");

      JMenuItem item = new JMenuItem("f2c");

      item.addActionListener(menuController);

      cmmdMenu.add(item);

      item = new JMenuItem("c2f");

      item.addActionListener(menuController);

      cmmdMenu.add(item);

      item = new JMenuItem("exit");

      item.addActionListener(menuController);

      cmmdMenu.add(item);

      menuBar.add(cmmdMenu);

   }
```

Finally, an inner controller class is declared:

```
   class MenuController implements ActionListener {
```

```
  private double getTemperature(String prompt) {

    String response = JOptionPane.showInputDialog(prompt);

    return new Double(response);

  }

  public void actionPerformed(ActionEvent e) {

    String cmmd = e.getActionCommand();

    if (cmmd.equals("f2c")) {

                    double fTemp = getTemperature("Enter Fahrenheit temperature");

      double cTemp = model.f2c(fTemp);

      celsiusField.setText("" + cTemp);

      farField.setText("" + fTemp);

    } else if (cmmd.equals("c2f")) {

      double cTemp = getTemperature("Enter Celisius temperature");

      double fTemp = model.c2f(cTemp);

      celsiusField.setText("" + cTemp);

      farField.setText("" + fTemp);

    } else { // cmmd == exit

      System.exit(0);

    }
    }

  }
```

LEARNING OUTCOME:

Our aim in this project was to create an Café Management System which would help the café owner to manage their Restraunts\Cafe in a structured manner. We successfully achieved that goal. Through the completion of this project we were able to learn a lot about a variety of java functions and classes, we faces some issues as well. After some issues we were able to finally achieve the desired result of reading inputs from the user and place the given order, edit the order and edit the menu as well. Getting to our end goal was not an easy nor was it a short process. We encountered issues in nearly all parts of the project. One of the first problems we encountered was accepting the inputs from the keyboard into readable numbers for the rest of the system to perform in well orderd way. We also encountered a couple of problems in displaying. The first of these problems was in that sometimes the orders would not display in the right places. While trying to solve this problem we discovered a latch in the code for the display. We solved this latch by changing the states where variables were not assigned new values in the same state that required them as an output.