UNIVERSITI TEKNOLOGI MARA

College of Computing, Informatics and Mathematics

CSC186 Object Oriented Programming

MINI PROJECT:
FINAL REPORT

Title: *Dato's Bakery Shop System*

Group: *CDCS1102A – Group 1*

Prepared by:

Student ID:	2022877512
Name:	NURIN IMAN BINTI MASNGOT
Student ID:	2022614994
Name:	SHAZWANA HUSNA BINTI SAARI
Student ID:	2022478924
Name:	MUHAMMAD AIDIEL BIN MOHAMAD HUSSIN
Student ID:	2022605596
Name:	MUHAMMAD NAZHAN BIN ROZAINI

Date Submitted:	1	4	0	7	2	0	2	3

Prepared for: MADAM NOR ZALINA BT ISMAIL

TABLE OF CONTENT

1.0 ORGANIZATIONAL STURCTURE

NURIN IMAN BINTI MASNGOT
(2022877512)
PROJECT MANAGER

MUHAMMAD AIDIEL BIN MOHAMAD HUSSIN
(2022478924)
PROGRAMMER 1

SHAZWANA HUSNA BINTI SAARI
(2022614994)
PROGRAMMER 2

MUHAMMAD NAZHAN BIN ROZAINI
(2022605596)
PROGRAMMER 3

2.0 INTRODUCTION TO PROJECT

Bakery self - ordering system have become a sought - out system nowadays. In handling busy life as a student, or a worker, some people don't have time to buy food, especially dessert in bakery shops. This is because most people are busy with their daily errands. Also, customers will rush their orders if they order at the counter, as they worry about holding up the line and causing other customers to run late, potentially causing mistakes in their orders to happen.

With the impact of COVID-19 still in the minds of our people, most people are still maintaining their efforts to social distance in closed and crowded areas. Our system helps them maintain their distance with our workers even while ordering their food. Besides, this self-ordering system (Dato's Bakery Shop) can reduce the time spent for employees to take a customer's order. This system will simply relay their orders to the staff. Thus, the efficiency of us within the bakery increases as it is less time - consuming. Besides, it is easy and user friendly when people use this system and making sure that customers can easily pick their orders without a rush.

Our bakery sells variety of cakes, ranging from simple spongy cakes to elaborate decorated cakes. We have multiple shapes and sizes our customers can pick from. Some examples of our cakes are Lemon Butterfly Cake, Shrek Green Tea Cake, Strawberry Night Sky Cake, Witchy Red Velvet Cake, Blue Ocean Cake, and Black Orchid Cake. Our customers may also enhance the taste of their acquired delicacies with the addition of several pastries of their choice such as brownies, croffle, donut, churros, tart, and bombolone with numerous flavours to decide on.

In this system, we prioritize the convenience of use to both our customers and the employees. For example, the system will ask customers to input the menu that they want. The system will read one output file and insert into system which is Order.txt, Pastry Order.txt, XX-XX-XXXX Orders.dat and Pastry Cheese Order.txt. After that, the system will calculate the total for each food, total for all orders, average price, highest price, and lowest price based on customers' orders. Furthermore, the system can be used by administrator to manipulate the process. Administrator can update and delete the input that customer enter to the system. Finally, administrator can edit customers phone number if customers give wrong number or changed the phone number.

3.0 OBJECTIVE

- To display the variety of cakes and pastries to the customer.
- To make it easy for the customer to order and customize their cakes and pastries.
- To attract people to use our system to ensure the satisfaction of customers.
- To apply and sharpen our understanding regarding the Object Oriented Programming aspect java coding.
- To increase efficiency, reduce errors and improve customers satisfaction.
- The systems should be user - friendly and easy to navigate for both employees and customers.

4.0 SCOPE

4.1 CLASS DIAGRAM

1. There are 4 classes: Bakery, Customer, Cake, and Pastry.

2. Customer and Bakery has a one-to-one relationship with composition link. It is a part of relationship where Bakery is the whole and Customer is the part-of objects. Bakery with same code can be purchased by one customer only or no customer at all. This implies a relationship which part of object Customer can't exist independently of the whole object Bakery.

3. Cake and Pastry are types of categories of Dato' Bakery's Shop System. This implies an inheritance relationship where Bakery has general attributes which is quantity, code, and composite customer. Cake has unique attribute which is the shape, weight, and radius while Pastry has unique attribute which is qtyPerBox, flavour, and addCheese.

4.2 USE CASE DIAGRAM

1. Only one actor is the Administrator. Both contract and permanent administrators can conduct the same process.

2. Process:

    a) Administrator can display menu.

    b) Administrator can separate orders to cake and pastry.

    c) Administrator can update customer phone number.

    Includes: Display confirmation message.
    Extends: Updates according to customer name.

    d) Administrator can add customer detail.

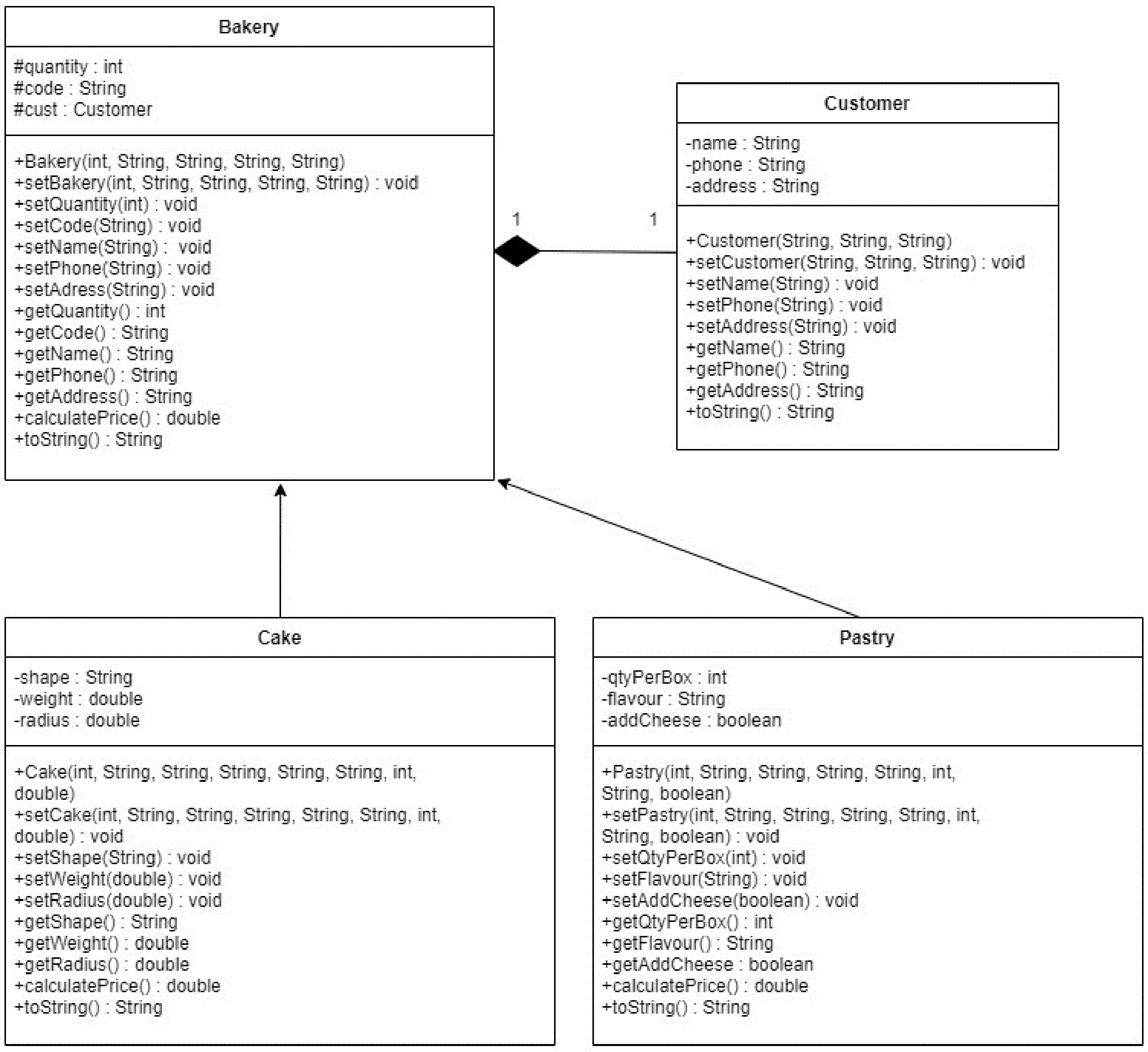
    e) Administrator can record daily orders.

    f) Administrator can filter for orders that choose to add cheese.
    Administrator.

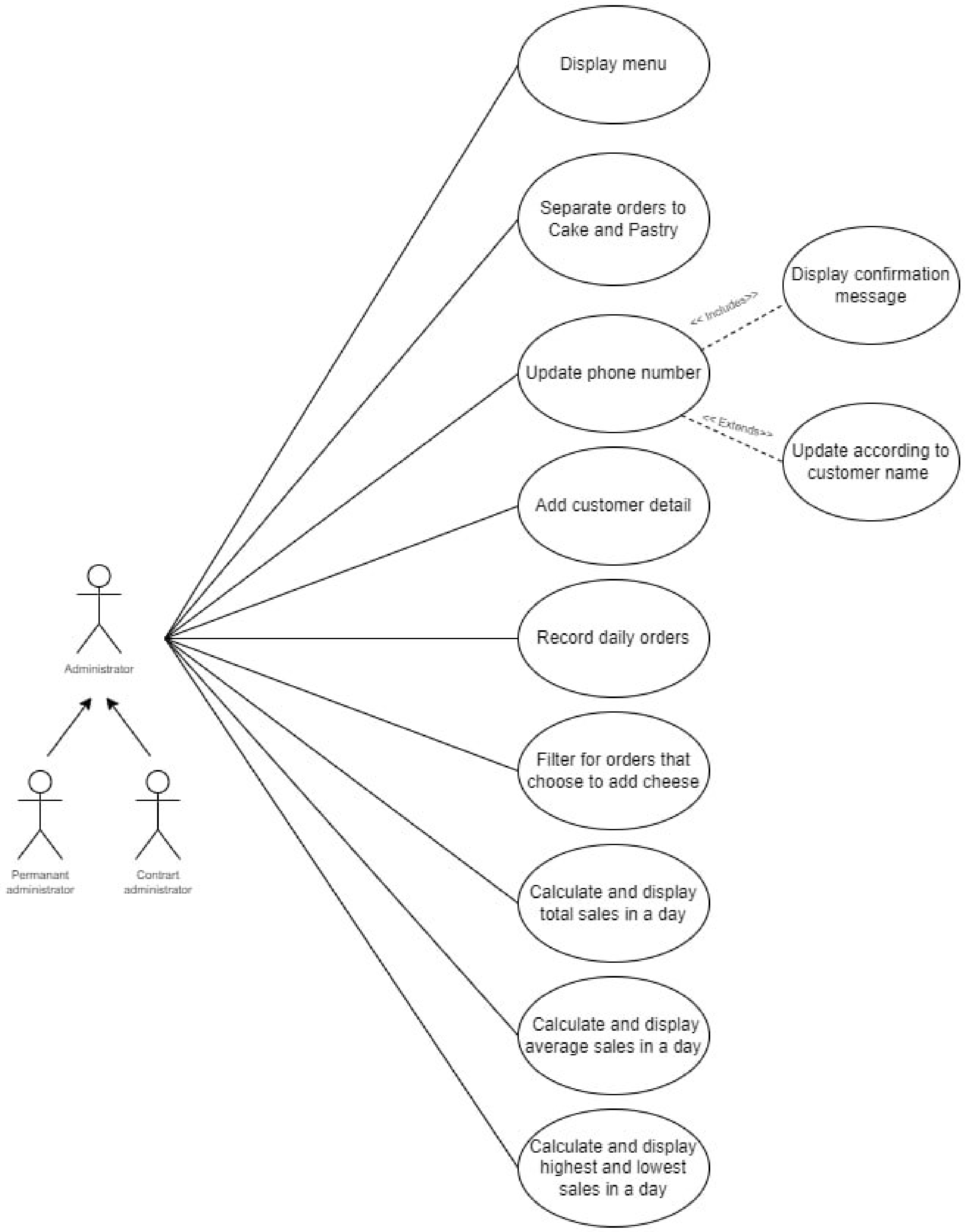
    g) Administrator can calculate and total sales, average sales, highest and lowest sales in a day.

5.0 UML DIAGRAM

5.1 UML CLASS DIAGRAM



5.2 UML USE CASE DIAGRAM


Display menu
Separate orders to Cake and Pastry
Display confirmation message
<< Includes>>
Update phone number
<< Extends>>
Update according to customer name
Add customer detail
Administrator
Record daily orders
Filter for orders that choose to add cheese
Permanant administrator
Contrart administrator
Calculate and display total sales in a day
Calculate and display average sales in a day
Calculate and display highest and lowest sales in a day

6.0 INPUT/OUTPUT FILE

6.1 INPUT FILE

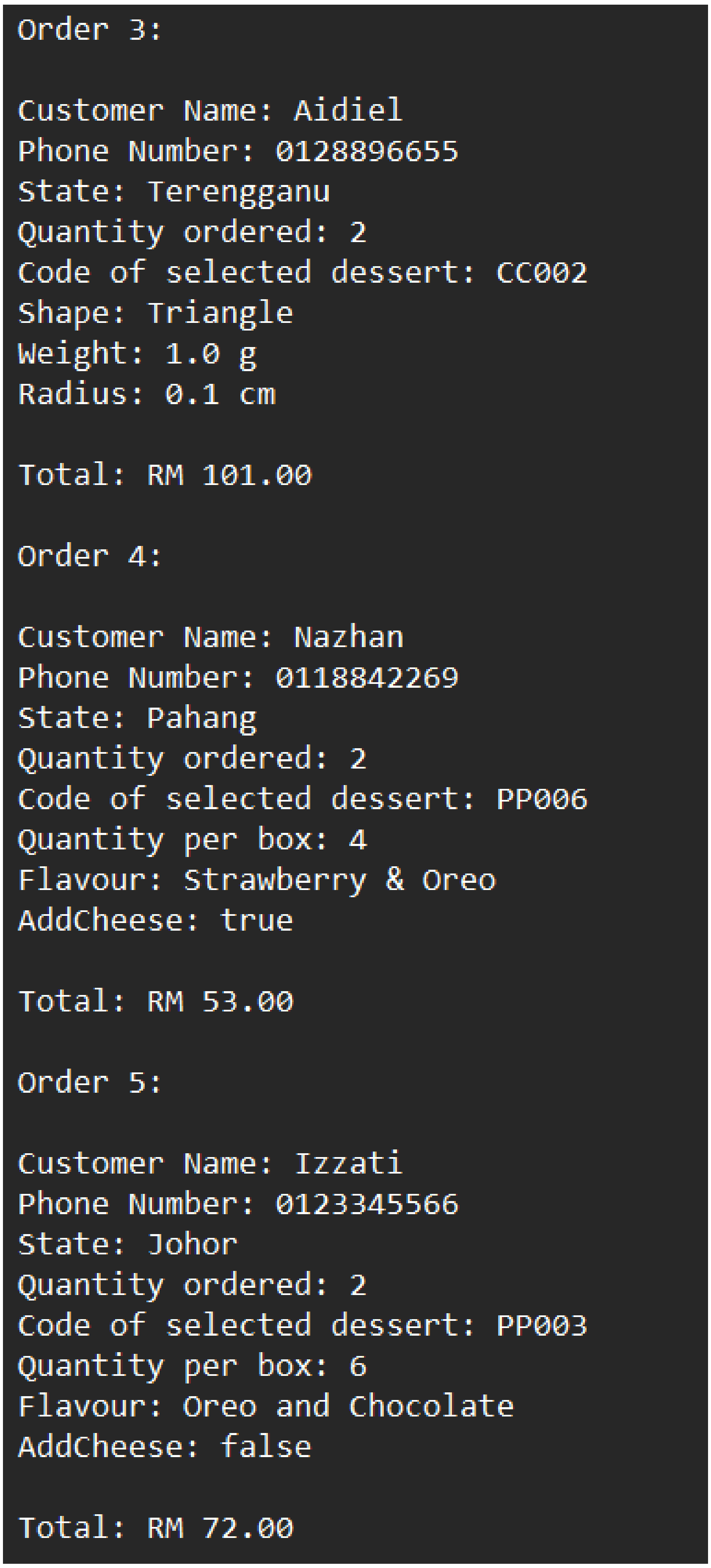
Order.txt

```
Nurin*0173669682*KL*CC002*1*Rectangle*1.5*0.1
Shazwana*0128896655*Seremban*PP003*2*4*Chocolate*true
Aidiel*0128896655*Terengganu*CC005*2*Triangle*1.0*0.1
Nazhan*0118842269*Pahang*PP006*2*4*Strawberry & Oreo*true
```

6.2 OUTPUT FILE

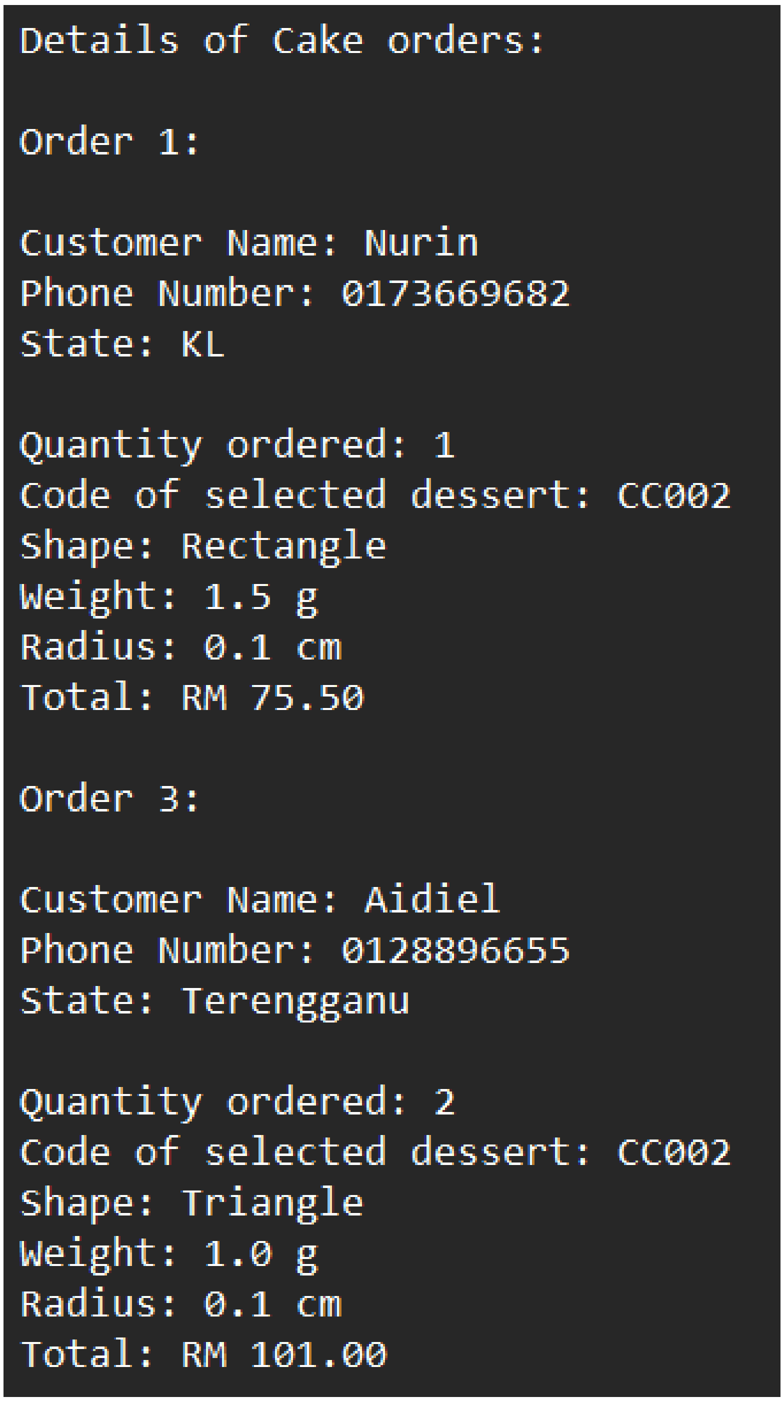
2023-07-12 Orders.dat

```
Order 1:

Customer Name: Nurin
Phone Number: 0173669682
State: KL
Quantity ordered: 1
Code of selected dessert: CC002
Shape: Rectangle
Weight: 1.5 g
Radius: 0.1 cm

Total: RM 75.50

Order 2:

Customer Name: Shazwana
Phone Number: 0128896655
State: Seremban
Quantity ordered: 2
Code of selected dessert: PP003
Quantity per box: 4
Flavour: Chocolate
AddCheese: true

Total: RM 53.00
```



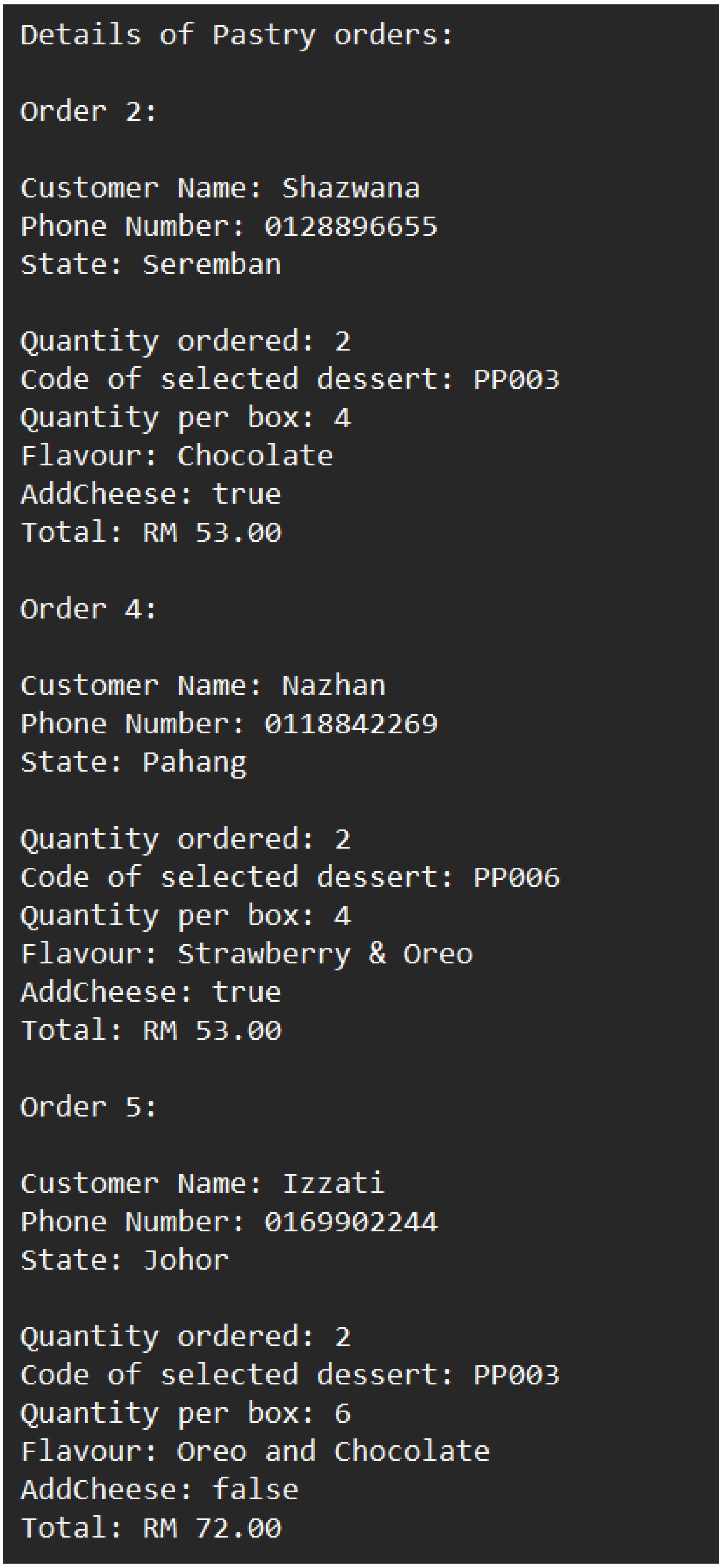
```
Order 3:

Customer Name: Aidiel
Phone Number: 0128896655
State: Terengganu
Quantity ordered: 2
Code of selected dessert: CC002
Shape: Triangle
Weight: 1.0 g
Radius: 0.1 cm

Total: RM 101.00

Order 4:

Customer Name: Nazhan
Phone Number: 0118842269
State: Pahang
Quantity ordered: 2
Code of selected dessert: PP006
Quantity per box: 4
Flavour: Strawberry & Oreo
AddCheese: true

Total: RM 53.00

Order 5:

Customer Name: Izzati
Phone Number: 0123345566
State: Johor
Quantity ordered: 2
Code of selected dessert: PP003
Quantity per box: 6
Flavour: Oreo and Chocolate
AddCheese: false

Total: RM 72.00
```

Cake Orders.txt



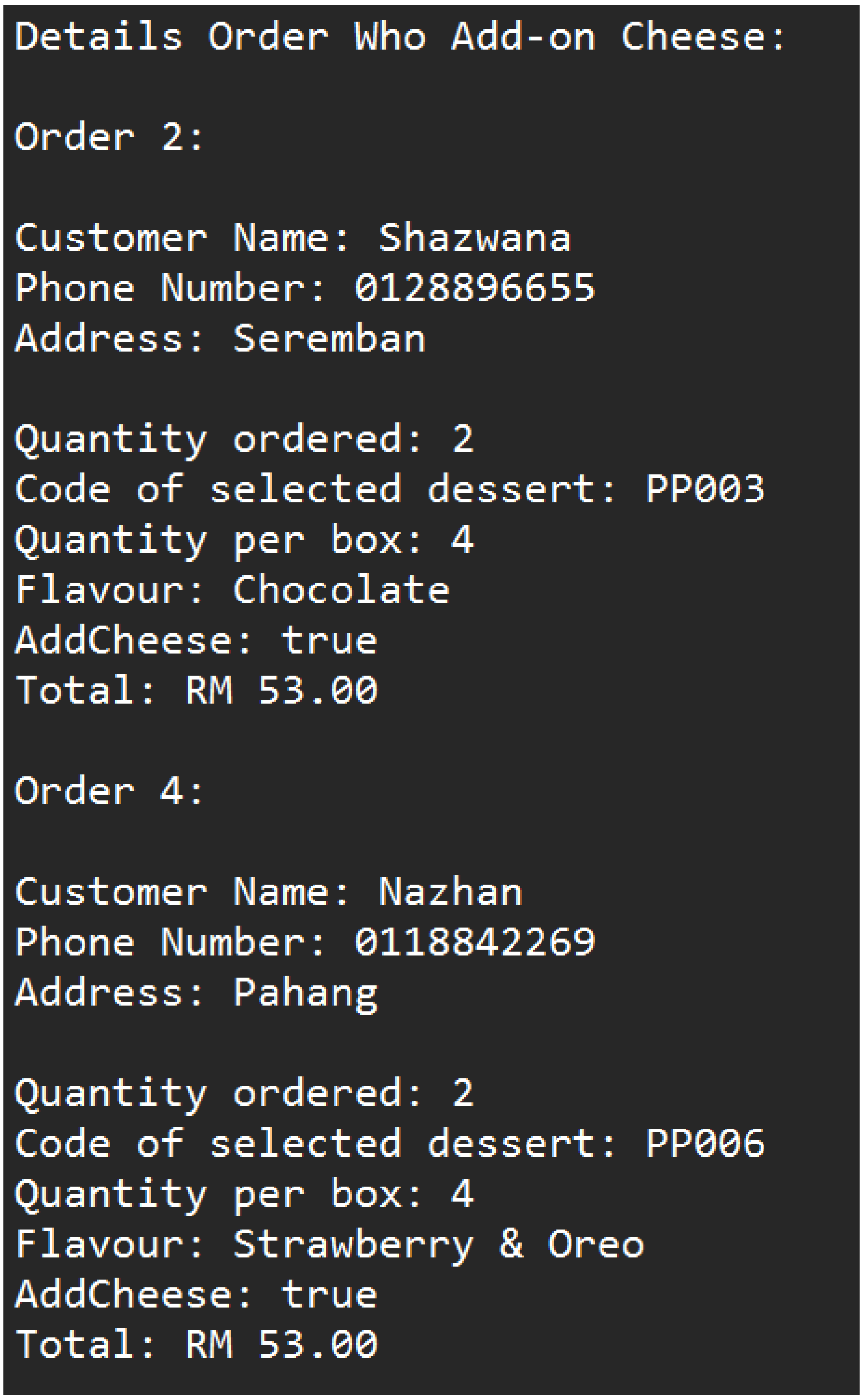
```
Details of Cake orders:

Order 1:

Customer Name: Nurin
Phone Number: 0173669682
State: KL

Quantity ordered: 1
Code of selected dessert: CC002
Shape: Rectangle
Weight: 1.5 g
Radius: 0.1 cm
Total: RM 75.50

Order 3:

Customer Name: Aidiel
Phone Number: 0128896655
State: Terengganu

Quantity ordered: 2
Code of selected dessert: CC002
Shape: Triangle
Weight: 1.0 g
Radius: 0.1 cm
Total: RM 101.00
```

Pastry Orders.txt



```
Details of Pastry orders:

Order 2:

Customer Name: Shazwana
Phone Number: 0128896655
State: Seremban

Quantity ordered: 2
Code of selected dessert: PP003
Quantity per box: 4
Flavour: Chocolate
AddCheese: true
Total: RM 53.00

Order 4:

Customer Name: Nazhan
Phone Number: 0118842269
State: Pahang

Quantity ordered: 2
Code of selected dessert: PP006
Quantity per box: 4
Flavour: Strawberry & Oreo
AddCheese: true
Total: RM 53.00

Order 5:

Customer Name: Izzati
Phone Number: 0169902244
State: Johor

Quantity ordered: 2
Code of selected dessert: PP003
Quantity per box: 6
Flavour: Oreo and Chocolate
AddCheese: false
Total: RM 72.00
```

Pastry Cheese.txt



```
Details Order Who Add-on Cheese:

Order 2:

Customer Name: Shazwana
Phone Number: 0128896655
Address: Seremban

Quantity ordered: 2
Code of selected dessert: PP003
Quantity per box: 4
Flavour: Chocolate
AddCheese: true
Total: RM 53.00

Order 4:

Customer Name: Nazhan
Phone Number: 0118842269
Address: Pahang

Quantity ordered: 2
Code of selected dessert: PP006
Quantity per box: 4
Flavour: Strawberry & Oreo
AddCheese: true
Total: RM 53.00
```

7.0 SOURCE CODE

//Superclass: Bakery

```
//This is superclass Bakery
public abstract class Bakery
{
   protected int quantity;                                 //declaration attribute quantity
   protected String code;                                  //declaration attribute code
   protected Customer []cust = new Customer[5];                  //declaration of array
for cust attribute

   //normal constructor
   public Bakery(int quantity, String code, Customer []cust)       //normal constructor
   {
      this.quantity = quantity;                              //quantity value accepted from
parameter as quantity attribute
      this.code = code;                                        //code value accepted from
parameter as code attribute
      for(int i = 0; i < cust.length; i++)                     //loop through the length of
cust array
         this.cust[i] = cust[i];                               //cust value accepted from
parameter as cust attribute
   }

   //setter method (group)
   public void setBakery(int quantity, String code, Customer []cust)  //setter method
(group)

   {

      this.quantity = quantity;                              //quantity value accepted from
parameter as quantity attribute

      this.code = code;                                      //code value accepted from
parameter as code attribute

      for(int i = 0; i < cust.length; i++)                   //loop through the length of
cust array

         this.cust[i] = cust[i];                             //cust value accepted from
parameter as cust attribute

   }



   //setter method (separate)

   public void setQuantity(int quantity){this.quantity = quantity; }   //set current
quantity's value

   public void setCode(String code){this.code = code; }              //set current code's
value

   public void setCust(Customer[] cust) {this.cust = cust;}          //set current cust's
value



   //getter method

   public int getQuantity(){return quantity; }                   //return the value of the
attribute quantity

   public String getCode(){return code; }                        //return the value of the
attribute code
```

```
   public Customer getCust(int loc){return cust[loc]; }              //return the value of the attribute cust



   public abstract double calculatePrice();                         //calculate price


   //print
   public String toString()
   {
      return("\nQuantity ordered: " + quantity +                    //return to display quantity
           "\nCode of selected dessert: " + code);                  //return to display code
   }
```

//Subclass: Cake

```
//This is a subclass called Cake
public class Cake extends Bakery
{
   private String shape;       //declaration attribute shape into String data type
   private double weight;      //declaration attribute weight into double data type
   private double radius;      //declaration attribute radius into double data type

   //normal constructor
   public Cake (int quantity, String code, Customer []cust, String shape, double weight, double radius)
```

```
    {
        super(quantity, code, cust);      //refer to the Bakery of a class or to call the
Bakery constructor
        this.shape = shape;        //initialize attribute shape into Cake class
        this.weight = weight;      //initialize attribute weight into Cake class
        this.radius = radius;      //initialize attribute radius into Cake class
    }

    //setter method for Bakery class
    public void setBakery(int quantity, String code, Customer []cust, String shape,
double weight, double radius)
    {
        super.setBakery(quantity, code, cust);      //refer to the Bakery of a class or to
call the Bakery setter method
        this.shape = shape;        //set new value for shape
        this.weight = weight;      //set new value for weight
        this.radius = radius;      //set new value for radius
    }

    //setter method for each Cake attributes
    public void setShape(String shape){this.shape = shape; }        //set new value for
shape
    public void setWeight(double weight){this.weight = weight; }      //set new value for
weight
```

```
  public void setRadius(double radius){this.radius = radius; }     //set new value for
radius


  //getter method
  public String getShape(){return shape; }           //return the value of the attribute
shape
  public double getWeight(){return weight; }         //return the value of the attribute
weight
  public double getRadius(){return radius; }         //return the value of the attribute
radius


  //calculation for Cake's Order
  public double calculatePrice()
  {
    double radPrice = 5.00; //price radius for 1centimetre
    double wPrice = 50.00; //price weight for 1kilogram


    double weightPrice = wPrice * weight;   //calculation for weight price (weight in
kg)
    double radiusPrice = radPrice * radius;     //calculation for radius price (price in
cm)


    double newPrice = (weightPrice + radiusPrice) * super.getQuantity();
//calculation of total price for Cake order
    return newPrice;    //return the value of the attribute newPrice
  }
```

```
    //toString() method

    public String toString()

    {

        return(super.toString() + "\nShape: " + shape +              //return to display shape
detail

                              "\nWeight: " + weight + " g"+     //return to display weight detail

                              "\nRadius: " + radius + " cm");   //return to display radius detail

    }

}
```

//Subclass: Pastry

```
//This is a subclass called Pastry

public class Pastry extends Bakery

{

    private int qtyPerBox;          //declaration of attribute qtyrPerBox

    private String flavour;        //declaration of attribute flavour

    private boolean addCheese;     //declaration of attribure addCheese


    //normal constructor

    public Pastry (int quantity, String code, Customer []cust, int qtyPerBox, String
flavour, boolean addCheese)

    {
        super(quantity, code, cust);      //refer to the Bakery of a class or to call the
Bakery constructor

        this.qtyPerBox = qtyPerBox;        //name value accepted from parameter as
qtyPerBox

        this.flavour = flavour;        //name value accepted from parameter as flavour

        this.addCheese = addCheese;       //name value accepted from paramater as
addCheese

    }


    //setter method for Bakery class

    public void setBakery(int quantity, String code, Customer []cust, int qtyPerBox,
String flavour, boolean addCheese)

    {

        super.setBakery(quantity, code, cust);       //refer to the Bakery of a class or to
call the Bakery setter method

        this.qtyPerBox = qtyPerBox;               //set the current qtyperBox's value

        this.flavour = flavour;                  //set the current flavour's value

        this.addCheese = addCheese;               //set the current addCheese's value

    }


    //setter method for each Pastry attributes

    public void setQtyPerBox(int qtyPerBox){this.qtyPerBox = qtyPerBox; }        //set
current qtyPerBox's value

    public void setFlavour(String flavour){this.flavour = flavour; }            //set current
flavour's value
```

```
   public void setAddCheese(boolean addCheese){this.addCheese = addCheese; }
//set current addCheese's value

   public void setCust(Customer[] cust) {super.setCust(cust);}              // set current
customer's value


   //getter method

   public int getQtyPerBox(){return qtyPerBox; }        //return value of the attribute
qtyPerBox

   public String getFlavour(){return flavour; }       //return value of the attribute flavour

   public boolean getAddCheese(){return addCheese; }      //return value of the
attribute addCheese


   //calculatePrice()

   public double calculatePrice()

   {

      double add = 0, qtyPrice = 6.00;  //set the price for each pastry


      //calculate the pastry based on qtyPerBox and qtyBox

      double pastryPrice = qtyPrice * qtyPerBox;


      if(addCheese == true)   //if the addCheese is true

         add = 5.00;        //add the price "RM5.00"

      else                  //else (addCheese=false)

         add = 0.00;        //add the price "RM0.00)
```

```
        double newPrice = (pastryPrice * super.getQuantity()) + add; //calculate the
total price

        return newPrice; //return the price

    }


    //toString() method

    public String toString()

    {

        return(super.toString() + "\nQuantity per box: " + qtyPerBox +  //return to display
qtyPerBox

                              "\nFlavour: " + flavour +           //return to display flavour

                              "\nAddCheese: " + addCheese);          //return to display
addCheese

    }

}
```

//Aggregation: Customer

//This class has composite relationship with Bakery, its function is to collect customer's information.

```
public class Customer

{

    private String name;                 //declaration of the attribute name (customer's
name)

    private String phone;                //declaration of the attribute phone (customer's
phone number)
```

```
    private String address;                //declaration of the attribute address (customer's
address)


    public Customer(String name, String phone, String address) //normal constructor

    {

        this.name = name;                  // name value accepted from parameter as name
attribute

        this.phone = phone;                // phone value accepted from parameter as phone
attribute

        this.address = address;            // address value accepted from parameter as
address attribute

    }


    //setter method

    public void setCustomer(String name, String phone, String address)        //setter
method for customer's details

    {

        this.name = name;                  //set the current name's value

        this.phone = phone;                //set the current phone's value

        this.address = address;            //set the current address' value

    }


    public void setName(String name){this.name = name; }                  //set current
name's value

    public void setPhone(String phone){this.phone = phone; }              //set current
phone's value
```

```
   public void setAddress(String address){this.address = address; }     //set current address' value


   //getter method
   public String getName(){return name; }             //return the value of the attribute name
   public String getPhone(){return phone; }           //return the value of the attribute phone
   public String getAddress(){return address; }      //return the value of the attribute address


   //display customers' details
   public String toString()
   {
      return("\nCustomer Name: " + name +     //return to display customer's name
           "\nPhone Number: " + phone +       //return to display customer's phone number
           "\nAddress: " + address);       //return to display customer's address
   }
}
```

//Main Application: DatosBakeryShopApp

```
import java.util.*;                    //load the contents of the java. util package
import java.io.*;                      //to use classes from the Java I/O (Input/Output) library
import java.time.LocalDateTime;              //to represents a date-time
import java.time.format.DateTimeFormatter;            //formatter for printing and parsing date-
time objects

public class DatosBakeryShopApp

{

   public static void main(String args[]) throws IOException //throws IOException for file
input/output

   {

      try{ //start try



      Bakery []order = new Bakery[5];                        //declaration for array of object order
into Bakery class

      Customer []cust = new Customer[5];                     //declaration for array of object cust
into Customer class



      //menu will display at console as customer's references



      System.out.println("Welcome to Dato's Bakery Shop!");                        //to greet user
of the console

      System.out.println("\n\t\t----------------------------------------");

      System.out.printf("\t\t\t\tMenu List");                                      // print header of the
console

      System.out.println("\n\t\t----------------------------------------");



      System.out.printf("\n\t\t\tCake:");                                          // print lists of cake
variations in the store

      System.out.printf("\n\t\t\tCC001 - Lemon Butterfly Cake");

      System.out.printf("\n\t\t\tCC002 - Shrek Green Tea Cake");
```

```
        System.out.printf("\n\t\tCC003 - Strawberry Night Sky Cake");

        System.out.printf("\n\t\tCC004 - Witchy Red Velvet Cake");

        System.out.printf("\n\t\tCC005 - Blue Ocean Cake");

        System.out.printf("\n\t\tCC006 - Black Orchid Cake");


        System.out.printf("\n\n\t\tPastry:");                              //print lists of pastry
variations in the store

        System.out.printf("\n\t\tPP001 - Brownies");

        System.out.printf("\n\t\tPP002 - Croffle");

        System.out.printf("\n\t\tPP003 - Donut");

        System.out.printf("\n\t\tPP004 - Churros");

        System.out.printf("\n\t\tPP005 - Tart");

        System.out.printf("\n\t\tPP006 - Bombolone \n\n");



        //system will input data based on file named "order.txt"

        FileReader fr = new FileReader("Order.txt");            //to open file named order.txt

        BufferedReader br = new BufferedReader(fr);              //buffer the input from the file


        int count = 0;                                   //initialize count into 0

        StringTokenizer st = null;                          //initialize token as null

        String dataRow = br.readLine();                      //to read one line of data


        while(dataRow != null)                              //to make sure read until the end of
data
```

```
    {   st = new StringTokenizer(dataRow, "*");                    //to cut the word based on "*"
delimiter


        String name = st.nextToken();                              //get next token as customer's
name

        String phone = st.nextToken();                             //get next token as customer's
phono number

        String address =st.nextToken();                            //get next token as customer's
address

        cust [count] = new Customer (name, phone, address);            //initialize customers'
details into Customer class


        String code = st.nextToken();                          //get next token as code from menu

        int quantity = Integer.parseInt(st.nextToken());           //get next token as quantity of
order


        //system will check the code if "CC" for cake OR "PP" for Pastry

        if(code.contains("CC"))                        //if code does contains 'CC' characters then
the statement block below will be executed

        {

           String shape = st.nextToken();                                          //get next token as
shape of the cake in order

           double weight = Double.parseDouble(st.nextToken());                           //get next
token as weight (kg) of the cake

           double radius = Double.parseDouble(st.nextToken());                           //get next
token as radius (cm) of the cake


           order[count]       =       new       Cake(quantity,code,cust,shape,weight,radius);
//initialize cake orders' details into Cake class
```

```
        }

        else if(code.contains("PP"))                    //however, if code contains 'PP' characters
instead then the statement block below will be executed

        {

            int qtyPerBox = Integer.parseInt(st.nextToken());                              //get next
token as quantity of pastry in each boxes

            String flavour = st.nextToken();                                          //get next token as
flavour of pastry

            boolean addCheese = Boolean.parseBoolean(st.nextToken());                         //get
next token as options to add cheese or not


            order[count] = new Pastry(quantity, code, cust, qtyPerBox,flavour, addCheese);
//initialize pastry orders' details into Pastry class

        }

        count++;                    //update the count variable

        dataRow = br.readLine();        //read the new line of data

    }



    //input from console

    Scanner input = new Scanner(System.in);                          //create Scanner object to
place input (int, double, float) from console into input

    Scanner inputText = new Scanner(System.in);                      //create Scanner object to
place input (String) from console into inputText


    //this condition executes if there's any blank elements in array
```

```
        for(int i = count; i < cust.length; i++)

        {

            System.out.println("Customer " + (i + 1) + " :");         //print header Customer and the
number of customer

            System.out.println("Enter Name: ");                   //prompt for customer's name

            String name = inputText.nextLine();                            //get customer's name from
Scanner object, inputText

            System.out.println("Enter Number Phone ");               //prompt for customer's phone
number

            String phone = inputText.nextLine();                       //get customer's phone number
from Scanner object, inputText

            System.out.println("Enter State: ");                 //prompt for customer's state

            String address = inputText.nextLine();                      //get customer's address from
Scanner object, inputText



            cust[i] = new Customer(name, phone, address);             //initialize customers' details
based on the inputs above into Customer class



            System.out.println("\nOrder Customer " + (i+ 1) + " :");       //print header Order
Customer and the number of the order

            System.out.println("Enter Code Bakery (Example: CCXXX - for cake /PPXXX - for
pastry): "); // Prompt for the code of the order

            String code = inputText.nextLine();                 //get customer's order from Scanner
object, inputText

            System.out.println("Enter Quantity: ");         //prompt customer for the quantity of their
order

            int quantity = input.nextInt();               //get the quantity through Scanner object input
```

```
	//using inputText code to determine if the order is to be put into Cake class or Pastry class

	if(code.contains("CC") || code.contains("cc"))		//if code contains the characters "CC" or "cc" regardless of case, then the statement block below will be executed

	{

	  System.out.println("Enter Shape (Rectangle/Triangle/Circle): ");	//prompt for the shape of the cake

	  String shape = inputText.nextLine();				//get the shape of the cake through Scanner object, inputText

	  System.out.println("Enter Weight Cake (in kilogram): ");		//prompt for the weight of the cake

	  double weight = input.nextDouble();				//get the weight of cake through Scanner object input

	  System.out.println("Enter Radius Cake (in centimetre): ");		//prompt to get radies of cake

	  double radius = input.nextDouble();				//get radius through Scanner object input


	  order[i] = new Cake(quantity, code, cust, shape, weight, radius);	//initialize cake order details into Cake class

	}


	else if(code.contains("PP") || code.contains("pp"))	//if code contains the characters "PP" or "pp" regardless of case, then the statement block below will be executed

	{

	  System.out.println("Enter Quantity Per Box (Maximum: 10): ");		//prompt for quantity of pastries in each box

	  int qtyPerBox = input.nextInt();				//get qtyPerBox through Scanner object input
```

```
            System.out.println("Enter Flavour Pastry (You can list more than 1 flavour as reference): ");       //prompt for the flavour of the pastry

            String flavour = inputText.nextLine();                                        //get flavour through Scanner object inputText

            System.out.println("Add-on Cheese? (true/false): ");                    //prompt for the options to add cheese to their pastries or not

            boolean addCheese = input.nextBoolean();                                  //get addCheese through Scanner object input


            order[i] = new Pastry(quantity, code, cust, qtyPerBox, flavour, addCheese); //initialize pastry orders' (input) details into Pastry class

          }

          System.out.println("\n");

      }



      //Output file: Cake Orders

      FileWriter fwC = new FileWriter("Cake Orders.txt");              // Write data fwC into a file (Cake Orders)

      BufferedWriter bwC = new BufferedWriter(fwC);                    // Buffer data for efficiet writing into the file

      PrintWriter pwC = new PrintWriter(bwC);                      // Print pwC as Cake file


      //to fitler order based on Cake and Pastry

      pwC.println("Details of Cake orders:");

      for (int i = 0; i < order.length; i++)             //system will loop through every order

      {
```

```
      if (order[i] instanceof Cake)            //if the order is from Cake class then the block
below will be executed
      {
        //Write their details into an output file named Cake Orders.txt
        Cake c = (Cake) order[i];                            //initialize order in Cake class
as 'c'
        pwC.println("\nOrder " + (i + 1) + ":");                    //print header according to
the number of the order
        pwC.println(order[i].getCust(i).toString());                //invoke customer's info to
be printed
        pwC.println(c.toString());
        pwC.printf("Total:  RM  %.2f",  c.calculatePrice());       //print  an  actual  amount
customer need to pay
        pwC.print("\n");
      }
    }


    //Output file: Pastry Orders
    FileWriter fwP = new FileWriter("Pastry Orders.txt");         //write data fwP into a file
(Pastry Orders.txt)
    BufferedWriter bwP = new BufferedWriter(fwP);                  //buffer data for efficiet
writing into the file
    PrintWriter pwP = new PrintWriter(bwP);                  //print data pwP for Pastry file


    //to fitler order based on Pastry
    pwP.println("Details of Pastry orders:");
    for(int i = 0; i < order.length; i++)          //s will loop through every order
    {

       if (order[i] instanceof Pastry)                    //if the order is from Pastry class then the
block below will be executed

       {

          //write the detail of them into an output file named Pastry Orders.txt

          Pastry p = (Pastry) order[i];                              //initialize order in Pastry class
as 'p'

          pwP.println("\nOrder " + (i + 1) + ":");                            //print header according to
the number of the order

          pwP.println(order[i].getCust(i).toString());                    //invoke customer's info

          pwP.println(p.toString());

          pwP.printf("Total: RM %.2f", p.calculatePrice());              //print and actual amount
customer need to pay

          pwP.print("\n");

       }

    }


    //Output file: Pastry Cheese Orders

    FileWriter fwCheese = new FileWriter("Pastry Cheese.txt");          //write data fwCheese
into a file (Pastry Cheese.txt)

    BufferedWriter bwCheese = new BufferedWriter(fwCheese);                //buffer data for
efficient writing into the file

    PrintWriter pwCheese = new PrintWriter(bwCheese);                  //print data pwCheese
for Pastry Cheese file


    //to filter order Pastry who order add cheese

    pwCheese.println("Details Order Who Add-on Cheese:");
    for (int i = 0; i < order.length; i++)            //system will loop through every order

    {

      if (order[i] instanceof Pastry)                  //if the order is from Pastry class then the
block below will be executed

      {

        Pastry p2 = (Pastry) order[i];            //initialize order in Pastry class as 'p2'

        if(p2.getAddCheese())

        {

          pwCheese.println("\nOrder " + (i + 1) + ":");          //print header according to the
number of the order

          pwCheese.println(order[i].getCust(i).toString());      //print invoke customer's info

          pwCheese.println(p2.toString());                    //print invoked customer's orders

        }

      }

    }



    //to calculate total sales and average sales in a day

    double sumAll = 0, average;                              //declare sumAll and average as
variables

    for(int i = 0; i < order.length; i++)                    //System will loop through every order

    {

      sumAll = sumAll + order[i].calculatePrice();              //sumAll will added with order[i]
prices in every loop

    }

    average = sumAll / count;                                //the average of all orders is sumAll
divided by count
```

```
        //to determine highest sales in a day

        double highest = order[0].calculatePrice();                    //declare highest as the price
from the first order

        for(int i = 0; i < order.length; i++)                 //system will loop through every order

        {

           if(order[i].calculatePrice() > highest)                     //if order's price is higher than
'highest'

              highest = order[i].calculatePrice();                //then the value inside highest will
be replaced with the current order's price

        }



        //to determine lowest sales in a day

        double lowest = order[0].calculatePrice();                     //declare lowest as the price
from the first order

        for(int i = 0; i < order.length; i++)                 //system will loop through every order

        {

           if(order[i].calculatePrice() < lowest)                      //if order's price is lower than
'lowest'

              lowest = order[i].calculatePrice();               //then the value inside lowest will be
replaced with the current order's price

        }



        //update phone number

        int found = -1, searchCount = cust.length;                             //declare found and
searchCount
```

```
    System.out.println("Confirm all customer's phone number were the latest one? (Change/Confirm): ");          //prompt to use the original or change customer's phone number

    String condition = inputText.nextLine();          //get condition through inputText

    for(int i = 0 ; i < searchCount; i++)          //System will loop through every customer

    {

      if(condition.equalsIgnoreCase("Change"))     //if condition is "Change", regardless of case,the statement block below will be executed

      {

        System.out.println("\nEnter Customer's Name: ");          //prompt for customer's name

        String searchName = inputText.nextLine();          //answer will be placed in searchName


        for(int j = 0; j < searchCount; j++)    //system will loop through every customer

        {

          if(order[j].getCust(j).getName().equalsIgnoreCase(searchName)) //if customer's name invoked is the same with searchName,

          {

            found = j;                    //then value of found will be replaced with the index of element with searchName

            break;

          }

        }


        if(found == -1)               //if the value of found is unchanged
```

```
            {
               System.out.print("\nThere's no records for customer name " + searchName + ". 
Try again");  //a message abt searchName non-existence will appear

               System.out.println("\nChange Another Customer's Phone Number? (Yes/No): ");
//prompt to change another customer's phone number

               String condition2 = inputText.nextLine();                    //answer will be 
placed in condition2


               if(condition2.equalsIgnoreCase("No"))             //if condition2 is 'No', then

                  break;                                   //the loop will break

            }


            else                          //however, if the value of found is not -1, then

            {

               System.out.println("Enter New Phone Number: ");      //prompt for new phone 
number will appear

               String newNumber = inputText.nextLine();          //the input will be placed inside 
newNumber

               order[found].getCust(found).setPhone(newNumber);      //the setPhone method 
from Customer class will be invoked to place the new value inside


               System.out.println("Change Another Customer's Phone Number? (Yes/No): ");
//prompt to change another customer's phone number

               String condition3= inputText.nextLine();                    //answer will be 
placed in condition2


               if(condition3.equalsIgnoreCase("No"))             //if condition2 is 'No', then
```

```
                break;                                  //the loop will break

            }

        }


        else if(condition.equalsIgnoreCase("Confirm")) //if condition is "Confirm"

            break;                              //then this segment will be skipped

    }



    //to output orders into a file with 'date' Orders.dat name

    //this is for employees to record daily orders to keep track of orders they need to fulfill

    FileWriter    fw    =    new    FileWriter(java.time.LocalDate.now()    +"    Orders.dat");
//write data fw into a file (__-__-____ Orders.dat)

    BufferedWriter bw = new BufferedWriter(fw);                                    //buffer data
for efficiet writing into the file

    PrintWriter pw = new PrintWriter(bw);                                        //p data pw for
Orders file



    for (int i = 0; i < order.length; i++)             //system will loop through every order

    {

        pw.println("\nOrder " + (i + 1) + ":");         //print header according to the number of the
order as pwP

        pw.println(order[i].getCust(i).toString());     //print invoked customer's detaild

        pw.println(order[i].toString());               //print details of recorded customers' orders

        pw.printf("\nTotal:  RM  %.2f",  order[i].calculatePrice());        //print  an  actual  amount
customer need to pay

    }
```

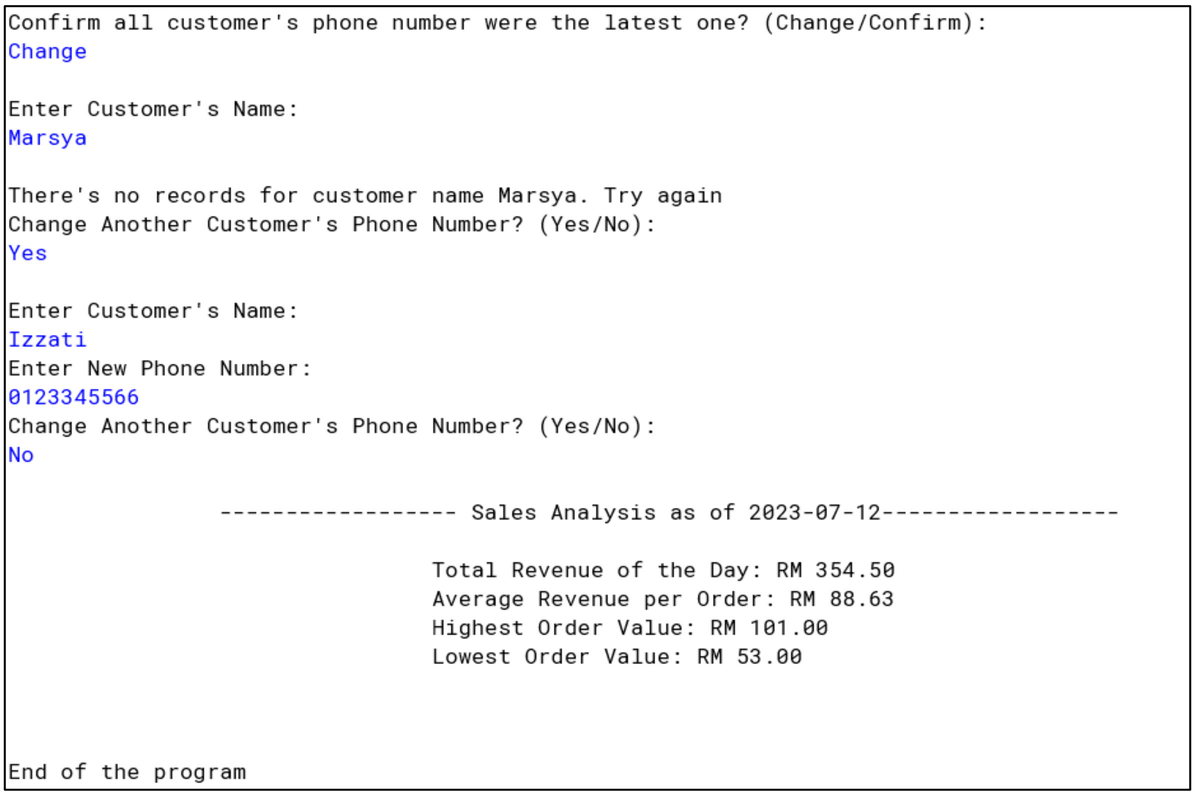
```
        //at the end of day, system executed sales analysis

        System.out.println("\n\t\t------------------    Sales    Analysis    as    of    "+
java.time.LocalDate.now()+ "------------------");    //header is about Sales analysis for the
current date

        System.out.printf("\n\t\t\t\tTotal Revenue of the Day: RM %.2f", sumAll);    //console
will display total sales

        System.out.printf("\n\t\t\t\tAverage Revenue per Order: RM %.2f", average);    //console
will display average sales

        System.out.printf("\n\t\t\t\tHighest Order Value: RM %.2f", highest);    //console will
display highest sale

        System.out.printf("\n\t\t\t\tLowest Order Value: RM %.2f", lowest);    //console will
display lowest sale



        br.close();        //close input file

        pw.close();        //close pw output file

        pwC.close();       //close pwC output file

        pwP.close();       //close pwP output file

        pwCheese.close();  //close pwCheese output file


       }//end try

        catch(EOFException eof)                //to display a message if an error
related to file occur

        {   System.out.println("\nProblem: " + eof.getMessage()); }   //display problem if the end
of the file or stream is reached unexpectedly

        catch(FileNotFoundException e)         //a file with the specified pathname
does not exist
```

```
        {     System.out.println("\nProblem: " + e.getMessage()); }       //display message of the
problem

        catch(IOException ioe)                                    //failed or interrupted I/O operations

        {    System.out.println("\nProblem: " + ioe.getMessage()); }   //to display message of the
problem

        finally

        {      System.out.println("\n\n\n\nEnd of the program"); }   //message displayed when
program ends

    }//end main

}
```

8.0 SAMPLE OUTPUT

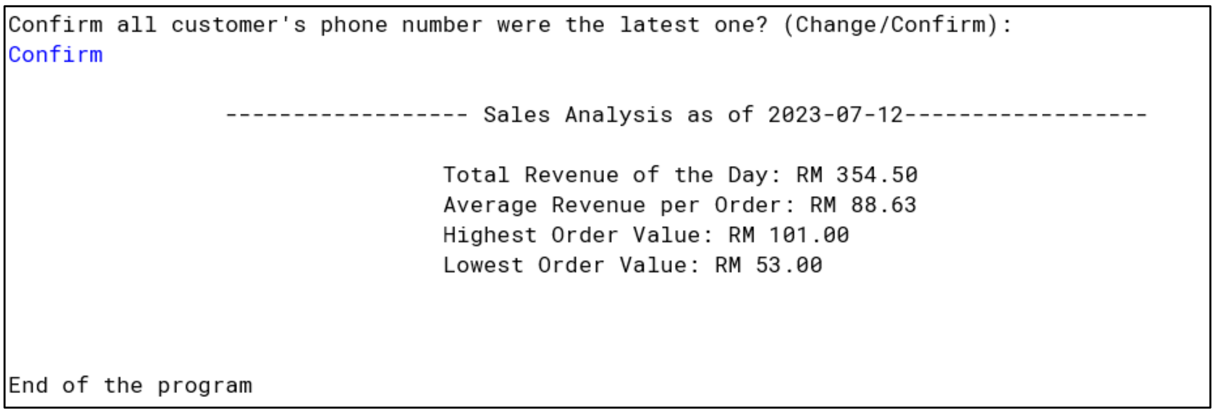
```
Welcome to Dato's Bakery Shop!

                  ----------------------------------------
                                Menu List
                  ----------------------------------------

                  Cake:
                  CC001 - Lemon Butterfly Cake
                  CC002 - Shrek Green Tea Cake
                  CC003 - Strawberry Night Sky Cake
                  CC004 - Witchy Red Velvet Cake
                  CC005 - Blue Ocean Cake
                  CC006 - Black Orchid Cake

                  Pastry:
                  PP001 - Brownies
                  PP002 - Croffle
                  PP003 - Donut
                  PP004 - Churros
                  PP005 - Tart
                  PP006 - Bombolone
```

```
Customer 5 :
Enter Name:
Izzati
Enter Number Phone
0169902244
Enter State:
Johor

Order Customer 5 :
Enter Code Bakery (Example: CCXXX - for cake /PPXXX - for pastry):
PP003
Enter Quantity:
2
Enter Quantity Per Box (Maximum: 10):
6
Enter Flavour Pastry (You can list more than 1 flavour as reference):
Oreo and Chocolate
Add-on Cheese? (true/false):
false
```



```
Confirm all customer's phone number were the latest one? (Change/Confirm):
Change

Enter Customer's Name:
Marsya

There's no records for customer name Marsya. Try again
Change Another Customer's Phone Number? (Yes/No):
Yes

Enter Customer's Name:
Izzati
Enter New Phone Number:
0123345566
Change Another Customer's Phone Number? (Yes/No):
No

                ------------------ Sales Analysis as of 2023-07-12------------------

                                Total Revenue of the Day: RM 354.50
                                Average Revenue per Order: RM 88.63
                                Highest Order Value: RM 101.00
                                Lowest Order Value: RM 53.00



End of the program
```

Note: If user choose "Change", system will ask to enter customer's name. If system didn't find that name, system will display error message and ask user if want to try again or not. If choose "Yes" system will repeat the process and if name was found, system will ask user to input new customer's phone number. Then, system will ask user if they want to change other customer's phone number or not. If user choose "Yes", system will repeat the same process, otherwise if user choose "No", system will skip the process and print the sales analysis on current date.



```
Confirm all customer's phone number were the latest one? (Change/Confirm):
Confirm

                ------------------ Sales Analysis as of 2023-07-12------------------

                                Total Revenue of the Day: RM 354.50
                                Average Revenue per Order: RM 88.63
                                Highest Order Value: RM 101.00
                                Lowest Order Value: RM 53.00



End of the program
```

Note: If choose "Confirm", system skipped the searching and update phone number's process and executed the sales analysis as current date.

9.0 REFERENCES

Farrel, J (2018). *Comprehensive Programming Logic & Design : Ninth Edition*. Retrieved January 15, 2023, from Programming Logic & Design, Comprehensive: Edition 9 by Joyce Farrell - Books on Google Play

D.S. Malik (2015). *C++ Programming: From Problem Analysis to Program Design : Seventh Edition*. Retrieved January 17, 2023, from C++ Programming: From Problem Analysis to Program Design - D. S. Malik - Google Books

Y. Daniel Liang (2014). Introduction to Programming with C++: Third Edition. Retrieved January 25, 2023, from Introduction to Programming with C++ 3rd INTERNATIONAL Edition, ISBN 13: 978-0273793243 | ebookschoice.com

Miller, Ronald E. (1984*). Input-Output Analysis : Foundations and Extensions*. Retrieved January 26, 2023, from Amazon.com: Input-Output Analysis: Foundations and Extentions: 9780134667157: Miller, Ronald E., Blair, Peter D.: Books

Gaddis, T (2016). *starting out with >>> Programming Logic And Design : Fourth Edition*. Retrieved January 27, 2023, from Starting Out with Programming Logic and Design: Gaddis, Tony: 9780133985078: Amazon.com: Books