**Business Type:** Ice Cream Parlor

**Company Name:** New Type Designers

**Pseudocode**

**Cost.java**

Within package pricecalc

Public interface Cost

{

*4 abstract public methods:*

addCost(double c)

String totalCost();

double showTax();

String showCost();

}

**Order.java**

Within package pricecalc

Import DecimalFormat to format prices

Public class Order implements Cost

{

*Local Declarations*

Double totalCost to keep track of costs

Double tax to apply tax

Decimal Format df to set format

}

Public Constructor Order(double c)

{

totalCost is assigned c

}

Public void addCost(double c)

{

totalCost is assigned itself + c

}

public String totalCost()

{

returns totalCost + (totalCost \* tax) formatted

}

public double showTax()

{

returns tax

}

public String showCost()

{

returns totalCost formatted

}

**CustomerOrder.java**

Within package parlororder

Import Scanner for input

Import package pricecalc for code

Class Store (Superclass)

{

*Protected Local Declarations*

String name of store

String address of Store

}

*Protected Methods*

Constructor Store(String n, String ad)

{

Name of store assigned n

Address assigned ad

}

showStore()

{

println Store Name

println Store Location

}

class IceCream extends Store (Subclass)

{

*Private Local Declarations*

String ice cream name

Int scoop amount

Double cost

}

*Methods*

Constructor IceCream(String n, String ad, String in, int sc, double c)

{

Call Constructor from superclass (n, ad)

Ice cream name assigned in

Scoops assigned sc

Cost assigned c

}

showIC()

{

println Ice Cream Name

println Scoop amount

println Cost

}

double getCost()

{

return cost

}

class CustomerOrder

main(String args[])

throws java.io exception

{

*Declarations*

Int choice

Char cChoice

Int flavor and scoops for temp storage

Double cost for temp storage

String name and add for temp store name storage

Scanner uChoice and chChoice, one for int input and one for char input

Do-while loop while choice != -1

{

Create new Order object

Output greeting to user and ask to choose from a set of stores

Stores and addresses

Get choice

Switch based on choice

Case 1-5

{

Temp store name assigned chosen store name

Temp store address assigned chosen store address

Output chosen store and ask for number of ice cream orders

limited from 1-5

While loop to make sure input is correct

Create new IceCream object array based on number inputted

For loop to iterate based on IceCream object array length

{

Choose flavor

Make sure input is correct

Temp flavor assigned flavor chosen

Choose scoop number, between 1-3

Make sure input is correct

Temp scoop assigned scoops chosen

Cost assigned number based on scoops chosen

Assign all values to IceCream array element based

on choices

Order Object.addCost(IceCream object element cost)

}

Output chosen store and Ice Cream orders chosen

Output subtotal, tax and total cost

Confirm order

If yes, output placed order and end program

If no, loop again

}

Case 6

{

Quit program

}

Default

{

If choice not found, ask user if they wish to try again or quit

If try again, reset loop again

Else quit

}

}

}

**Documentation**

The program is named Ice Cream-4-U, created to order ice cream from various stores. The program’s main purpose is to make it easy to order from any ice cream parlor in the program’s system instead of using an individual system for each store. It is because it would be easier to help the smaller stores if they were all within a single system for a user to use rather than have to set-up their own system themselves. The programs sub-purpose is to make it easy to order ice cream through a couple of inputs on their own time rather than having to enter the parlor to find out what they have.

The program is designed for both businesses and customers. This program is designed for businesses because it allows for them to be inputted into the system with their inventory and allows them to take orders from customers without having to wait for them to arrive. It also assists businesses in that all they need do is opt into the system and it will allow them to be found by anyone using the program. The program also assists customers because it allows them to quickly find an ice cream parlor and place an order at that location. This makes it quicker for them to select what they want instead of having to go the parlor and use more time from their day to decide what kind of ice cream they want.

The program’s business function is to make it easier to choose a parlor to make an order at. By knowing the inventory and the parlor’s information, the business can easily be sent an order once the infrastructure is set-up. This will allow it to be more noticeable to the public than if it simply relied on location itself.

The program’s technical function is to store a customer’s order, based on the chosen parlor, and then give them the total once the order is placed. By creating an object array, the order can be placed and added to based on the customer’s desires and choices. This makes it easy for the customer to choose the various options within the program without having to deal with convoluted inputs. By the end, the program will output all of the information needed for the customer to place the order for their ice cream, knowing how much it will cost with and without tax, as well as showing the user what they have ordered.