**CENTENNIAL COLLEGE**

**COMP 228: JAVA PROGRAMMING**

**LAB TEST 1: Version 2**

**(For Students whose Last Name (or First Name, if You Don’t Have a Last Name) Starts with the letter from N to Z)**

**Full Marks: 100**

**Time allowed: 3 hours**

**NAME: Aryan Patel (301226774)**

**Be sure to read the following general instructions carefully:**

**This lab should be completed individually by all the students.**

YOU NEED TO SUBMIT THE FOLLOWING 2 DOCUMENTS IN THE DROPBOX TITLED MidTermTest:

1. THE FIRST ONE IS A WORD DOCUMENT. USE THIS DOCUMENT AND ADD SCREEN SHOTS OF THE RUNNING STATE OF THE APPLICATION. DO NOT DELETE THE QUESTIONS. THE SCREEN SHOTS SHOULD COVER ALL THE ASPECTS/FUNCTIONALITIES OF THE APPLICATION. AFTER THE SCREEN SHOTS PLEASE COPY THE CODE FROM THE CODE WINDOW AND PASTE THE COMPLETE CODE INTO THE SAME WORD DOCUMENT. DO NOT GIVE ME SCREEN SHOTS OF THE CODE. DO NOT ZIP THIS FILE AND KEEP IT SEPARATE FROM YOUR ZIPPED PROGAM FILE.

2. SUBMIT ALSO ONE ZIPPED PROJECT/PROGRAM FILE THAT CONTAINS THE PROJECT IN ITS ENTIRITY SEPARATELY INTO THE SAME DROP BOX.

You must name your Eclipse/IntelliJ project and your Word file according to the following rule:

YourFullName\_COMP228MidLabTest

Example: JohSmith\_COMP228MidLabTest

Apply the naming conventions for variables, methods, classes, and packages:

- variable names start with a lowercase character

- classes start with an uppercase character

- packages use only lowercase characters

- methods start with a lowercase character

**General Scenario:**

You are creating an interactive CONSOLE application (NOT a GUI application) that gets input from the keyboard and keeps track of a grocery store that a businessman operates. To develop this application, you will be using TWO (2) interfaces, ONE (1) abstract class, and ONE (1) concrete class. You should use the TWO interfaces and the ONLY abstract class to create the concrete class. Your application should have overridden toString() at appropriate places so that you could display the object state, i.e., the string representation of the objects that you create by simply using the following command: System.out.println(object).

**Follow the following instructions:**

1. Create 2 interfaces: Onion and Coke.

Interface Onion contains only 1 abstract method: onionPrice() which does not return any data. It takes 1 double argument: onionPrice that represents the current price (per kilo) of onion.

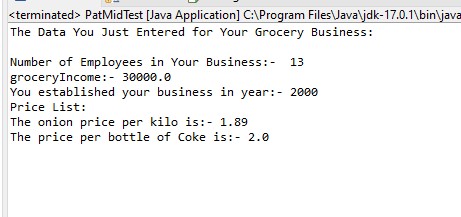
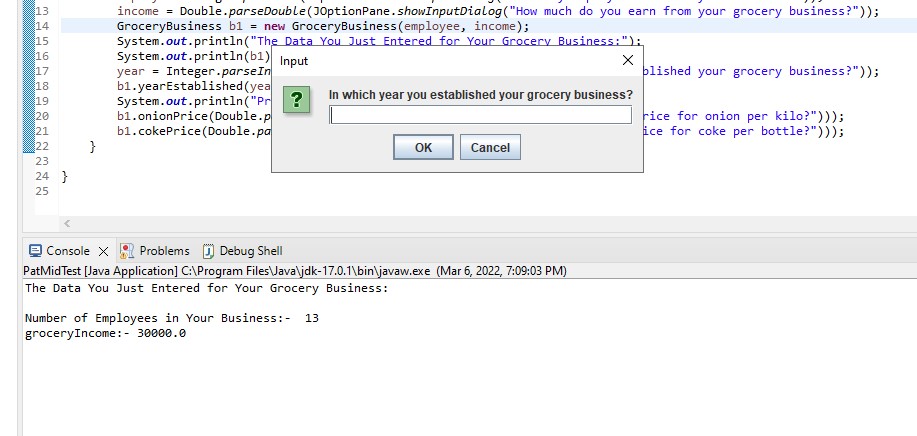
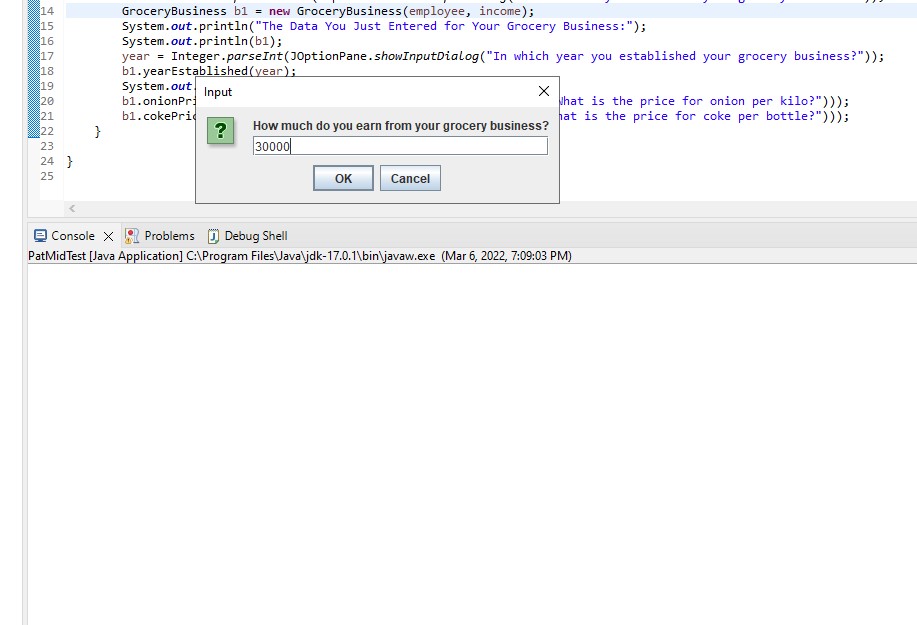
Interface Coke contains only 1 abstract method: cokePrice() which does not return any data. It takes 1 double argument: cokePrice that represents the current price (per bottle) of coke.

1. Create an abstract class named Businsess. This class contains 1 instance variable, 1 constructor, and 1 abstract method.
2. Instance variable: numEmp (that will take care of the number of employees in the business. Use appropriate data type.
3. A 1-argument constructor that will be called from its subclass to load 1 instance variable of this level during the creation of an object in its subclass.
4. 1 abstract method: numEmployee(). This method does not take any arguments. When implemented in its subclass, it will return an integer that represents the current number of employees currently working for the business.
5. In this abstract class Business, write necessary code to get the object state (i.e., the string representation) that will eventually be called from its concrete subclass. **[25 marks]**
6. Create a concrete class named GroceryBusiness that makes use of both the interfaces: Onion and Coke that you have created earlier. This concrete class also makes use of the abstract class Business that you have also created earlier.
7. a. This concrete class GroceryBusiness has 1 instance variable of its own, 1 constructor, and 1 concrete method of its own.
8. Instance variable: groceryIncome (Use appropriate data type) that represents the current yearly income of the grocery business.
9. Constructor: that invokes the constructor of its parent class and eventually constructs a GroceryBusiness object with the instance variables it needs to construct the GroceryBusiness objcct.
10. 1 concrete method of its own: yearEstablished() that does not return any data. It takes 1 integer argument (year) that represents the year the grocery business was established. This method displays the year the grocery business was established from within itself.
11. Write ALL OTHER necessary code to make this class a concrete class.
12. Write necessary code to get the complete string representation (i.e., the object state) of objects made from this class.

**[35 marks]**

1. Create a driver class named XXXMidTest (replace XXX by three letters of your last name, if you don’t have a last name, use your first name). This driver class would prompt the user to enter current number of employees in your grocery business. It would then ask how much you earn from the grocery business. It then uses these 2 user input to create a GroceryBusiness object. It would then display the string representation of the object you just created. Then it would ask for the year the grocery business was established.. It would display the year you just entered. It would then ask for the current price of onion per kilo. And finally it would ask for the current price of coke per bottle. It would then display the prices you just entered.

**[40 marks]**

****

**Interfaces:**

**package** exercise1;

**public** **interface** Onion {

**public** **abstract** **void** onionPrice(**double** price);

}

**package** exercise1;

**public** **interface** Coke {

**public** **abstract** **void** cokePrice(**double** price);

}

**Abstract:**

**package** exercise1;

**public** **abstract** **class** Businsess {

**public** **int** numEmp;

**public** Businsess(**int** numemp) {

**this**.numEmp = numemp;

// **TODO** Auto-generated constructor stub

}

**public** **abstract** **int** numEmployee();

@Override

**public** String toString()

{

**return** "\nNumber of Employee:- " + **this**.numEmp;

}

}

**Concrete class:**

**package** exercise1;

**public** **class** GroceryBusiness **extends** Businsess **implements** Coke, Onion {

**public** **double** groceryIncome;

**public** **double** onionPrice;

**public** **double** cokePrice;

**public** **int** year;

**public** GroceryBusiness(**int** numemp, **double** gIncome) {

**super**(numemp);

**this**.groceryIncome = gIncome;

// **TODO** Auto-generated constructor stub

}

**public** **void** yearEstablished(**int** year)

{

System.***out***.println("You established your business in year:- "+year);

**this**.year = year;

}

@Override

**public** **void** onionPrice(**double** price) {

// **TODO** Auto-generated method stub

**this**.onionPrice = price;

System.***out***.println("The onion price per kilo is:- "+price);

}

@Override

**public** **void** cokePrice(**double** price) {

// **TODO** Auto-generated method stub

System.***out***.println("The price per bottle of Coke is:- "+price);

**this**.cokePrice = price;

}

@Override

**public** **int** numEmployee() {

// **TODO** Auto-generated method stub

**return** **this**.numEmp;

}

@Override

**public** String toString()

{

**return** "\nNumber of Employees in Your Business:- " + **this**.numEmp +

"\ngroceryIncome:- "+**this**.groceryIncome;

}

}

**A possible test run is show below: (Give me ONE Screen shot with input entered and output produced, as shown below). DO NOT USE the same data. Use your personalized data.**

How many employees work in your business?

12

How mmuch do you earn from your grocery business?

120000

The Data You Just Entered for Your Grocery Business:

Number of Employees in Your Business: 12

groceryIncome: 120000

In which year you established your grocery business?

1999

You established your business in year: 1999

What is the price for onion per kilo?

1.29

What is the price for coke per bottle?

2.49

Price List:

The onion price per kilo is: 1.29

The price per bottle of Coke is: 2.49