Phase 5 IP

IT251-1402A-01: Intermediate Java Programming I

Jeffery Sanford

Colorado Technical University

Tim Paulson

May 12, 2014

Contents

[Phase1 3](#_Toc387696690)

[Phase 2 5](#_Toc387696691)

[Phase 3 8](#_Toc387696692)

[Final Design Document Phase 4 10](#_Toc387696693)

[Phase 5 Final 15](#_Toc387696694)

[UML Class Diagram: 15](#_Toc387696695)

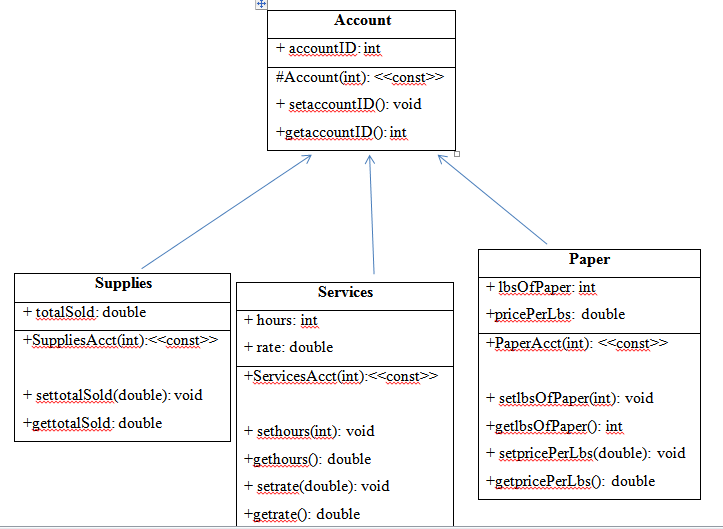
[Pseudocode: 16](#_Toc387696696)

[Runtime Screenshots: 17](#_Toc387696697)

[Competency Test 19](#_Toc387696698)

[References 20](#_Toc387696699)

# Phase1



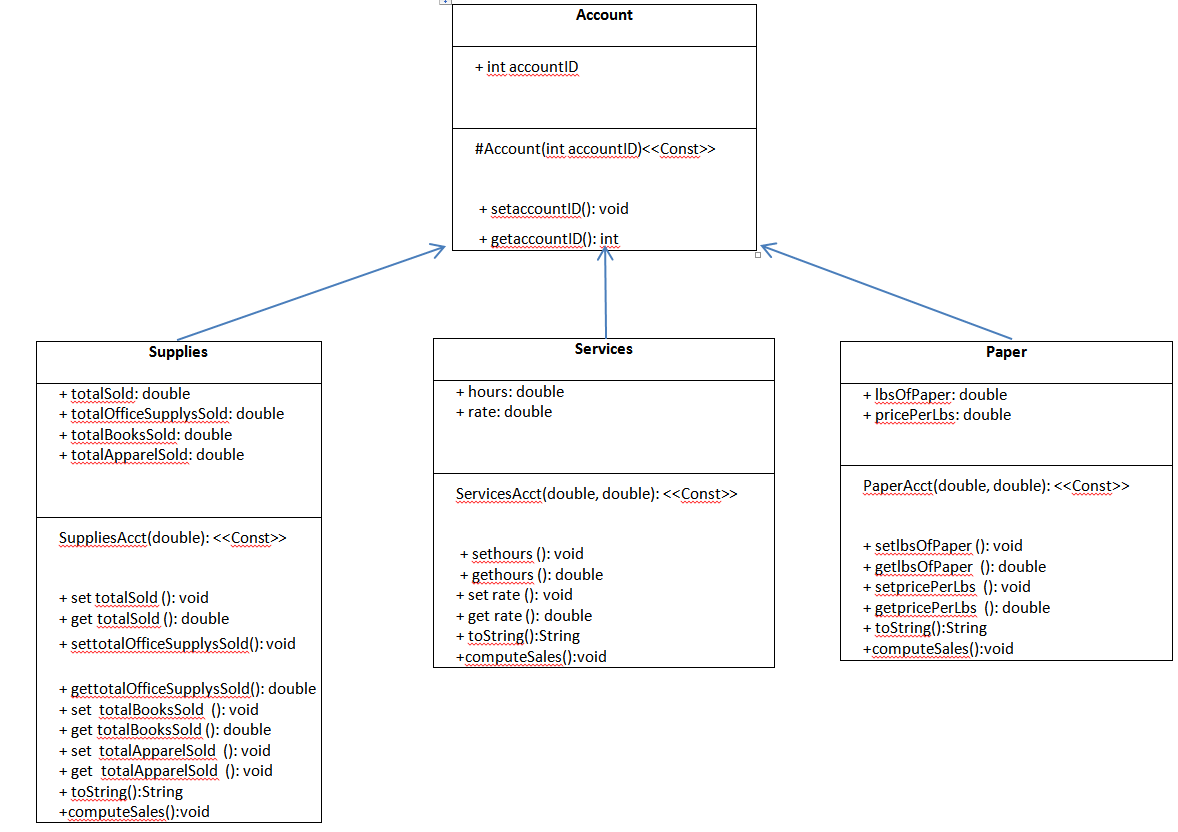
In this assignment the task was to create a superclass with three subclasses along with their individual attributes. As you can see from the UML class diagram above the Class Account is the super class with one attribute, accountID. There are also two methods within this class, a set and get method. The other three classes are the subclasses which are extended from the Account class. By extending the three subclasses from the Account class each subclass will inherit the super classes members. The arrows pointing to the Account class indicate the subclasses extend from the Account class.

Writing code to incorporate an inheritance hierarchy allows for code to be reused in a program. The best way to understand how this happens is to understand the definition of inheritance in computer programming. Whenever a class is extended from another class it inherits all the members from that class. This is known as inheritance. Therefore if you class “b” extends “a” and class “c” extends “b” then both b and c inherit from class a. Class c will be able to reuse all the code from class a and b ("Java Inheritance", n.d., p.1).

Most computer programs will have several thousand lines of computer code, if not more. It is beneficial for the programmer to implement and inheritance hierarchy in order to make the application easier to maintain. As mentioned above if code re-usage is implemented properly by way of an inheritance hierarchy then all classes and their members will be easier to maintain. If a programmer implements inheritance in his or her program and a piece of code must be manipulated in the super class then that one change will be filtered down to all extended subclasses. In the case of this class assignment adding an attribute to the Account class will then give all other extended subclasses access the that member.

In the future, if management would like for another account type to be added to the design above the following steps would need to be taken. First, a new subclass of the Account class would need to be extended so that it is consistent with the inheritance hierarchy structure. Second, the new class would then need to be defined further with customized attributes that need to be tracked ("The UML Class Diagram: Part 1 - Developer.com - Developer.com", n.d., 2003).

# Phase 2



As you can see from the class diagram above the classes have changed a little now that there is a better understanding of the project requirements. The only attribute at this point that is needed for the superclass is the accountID. The Account superclass has the typical set and get methods as well. Per the new requirements of this discussion board three new attributes have been added to the Supplies class. These new attributes, totalOfficeSupplysSold, totalBooksSold and totalApparelSold will be used in the computeSales method, which has been added to each of the subclasses. This new method will be used to compute the total sales of each account type. You may notice that the overloaded computeSales method is not in the superclass. The reason for this is that the account class will be declared as an abstract class. This means the superclass will only be providing inherited members to the subclasses. The computeSales class will be overloaded and placed in each subclass to provide a method for specific calculations for that class ( Abstract Methods and Classes).

Below you will find the pseudocode for IP2’s requirements.

Main()//main method

Ask for sales person’s name

Get name

Ask for supplies account id

Get id

Ask for amt of supplies sold

Get amt

Ask for amt books sold

Get amt

Ask for apparel sold

Get amt

Ask for services account id

Get it

Ask for service hours

Get hour

ask for rate

Get rate

Ask for paper account id

Get id

Aks for lbs of paper sold

Get lbs

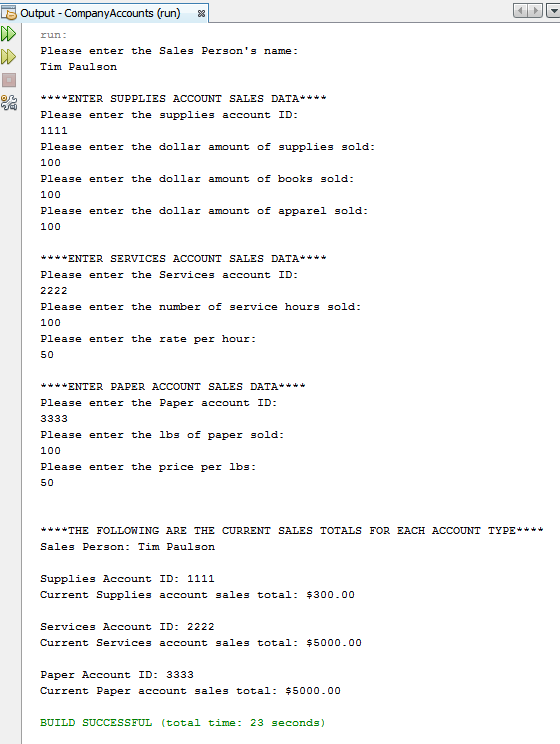
Ask for price per lbs

Get lbs

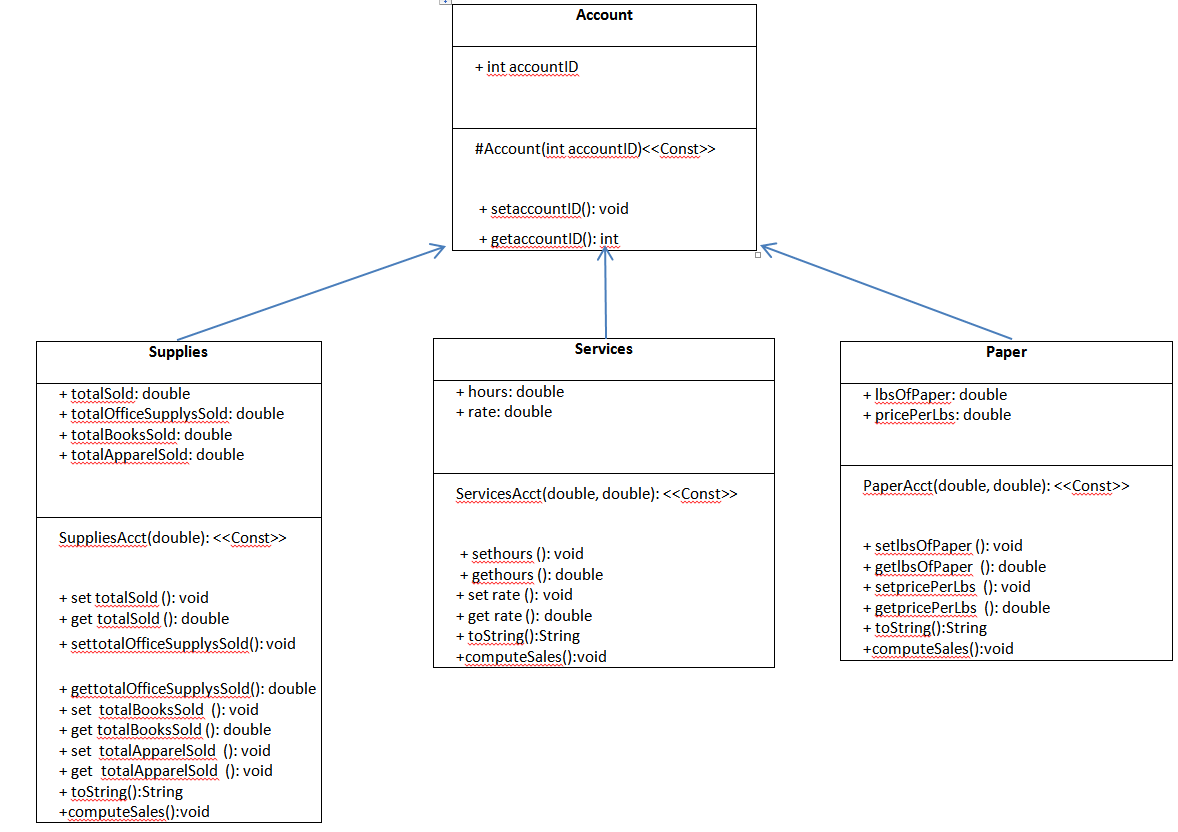
display current sales totals for all accounts

exit program

The following screenshot is a demonstration that the program runs and compiles free of errors.



# Phase 3



In this assignment not much has changed from last phase. However, the main method was changed to incorporate encapsulation. The following is a pseudocode representation of my main method.

Main()

Welcome user to program

Ask user for their name

Get name

getSuppliesAcctSalesData();// this method and the following incorporates encapsulation

getServicesAcctSalesData();

getPaperAcctSalesData();

instantiate Supply, Services and Paper account objects

assign objects instantiated to accountArray[3] index value locations

Display summary message title

Display sales persons name

Enter for loop

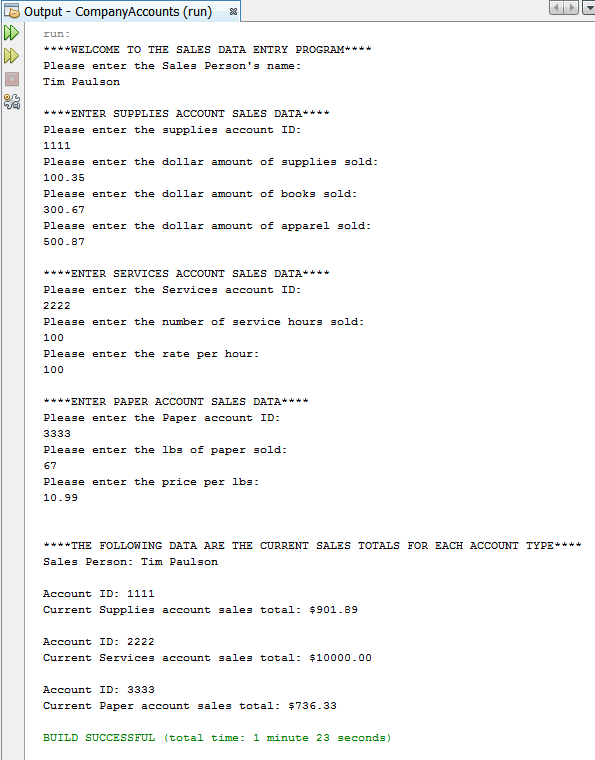
accountArray[i].computeSales()

print accountArray[i]

exit for loop after all objects have been printed

end main()

The following screenshot is proof that my program runs and compiles without error.

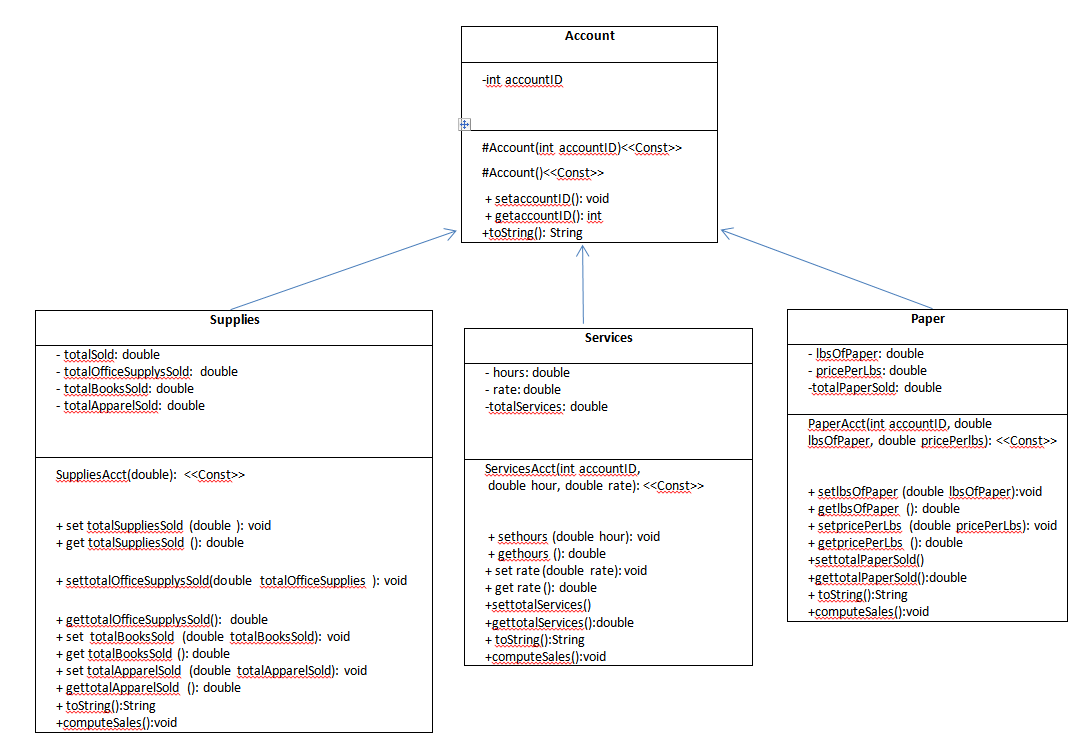


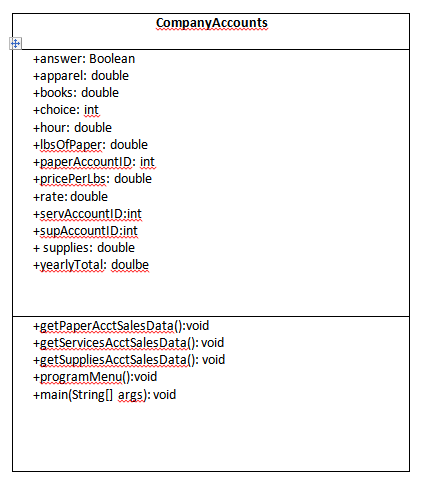
# Final Design Document Phase 4

The following program allows the company’s sale employees (SA) to enter sales data for the company’s three account types, Supplies, Services, and Paper accounts. The program will display a welcome message and a menu with five choices. Choice 1-3 will allow the SA to enter all sales data. At any given time the SA may select option 4 to calculate and display the totals for each account as well as the yearly total. Option 5 allows the user to exit the program.

The following are the class diagram, pseudocode, and program screenshots captured at runtime.

UML Class Diagram:





Pseudocode:

Main(){

Instantiate 3 account types

Display welcome

Try{

While{

Display program menu

User inputs option number

Switch{

Case 1 getSuppliesAcctSalesData()

Case 2 getServicesAcctSalesData()

Case 3 getPaperAcctSalesData()

Case4 instatiate array with account types

For loop computes and displays account totals

Yearly total is calculated and displayed

Case 5 terminates the program

Default if the user enters a number other than in the range of 1-5

It will display an error message

}//end switch

}//end while

}//end try

Catch {

Catches the InputMismatchExeption if the user enters anything other than a number

Displays error message and Terminates the program

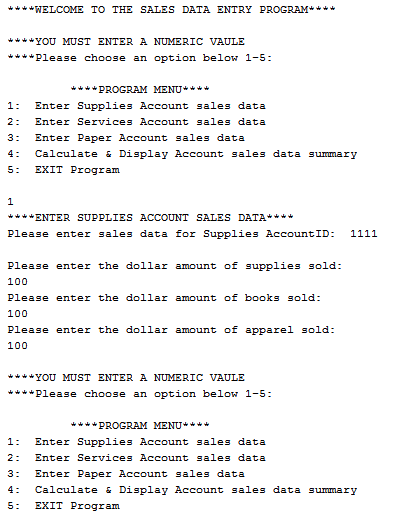
}//end catch

}//end main()

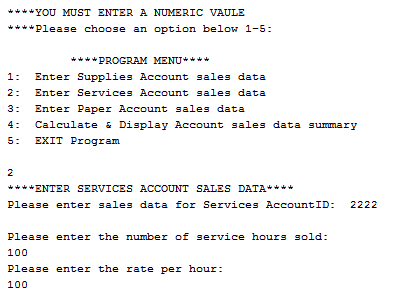
Runtime Screenshots:

Welcomes the user

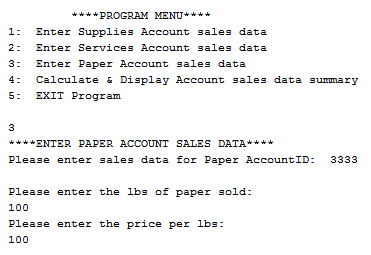
User enters option 1 and enters values



User chooses option 2 and enters values

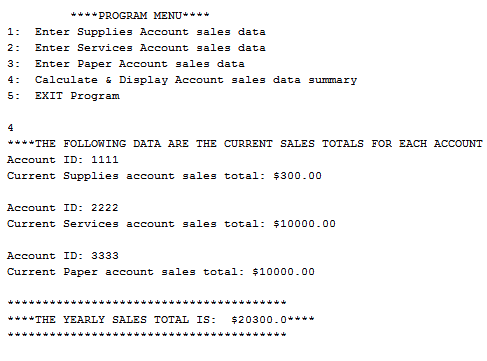


User enters option 3 and enters values

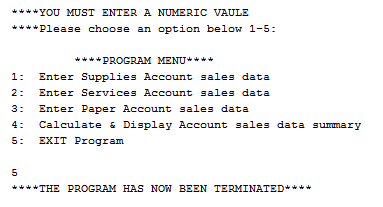


User enters option 4

Program calculates account totals and displays yearly total



User enters option 5 and the program terminates

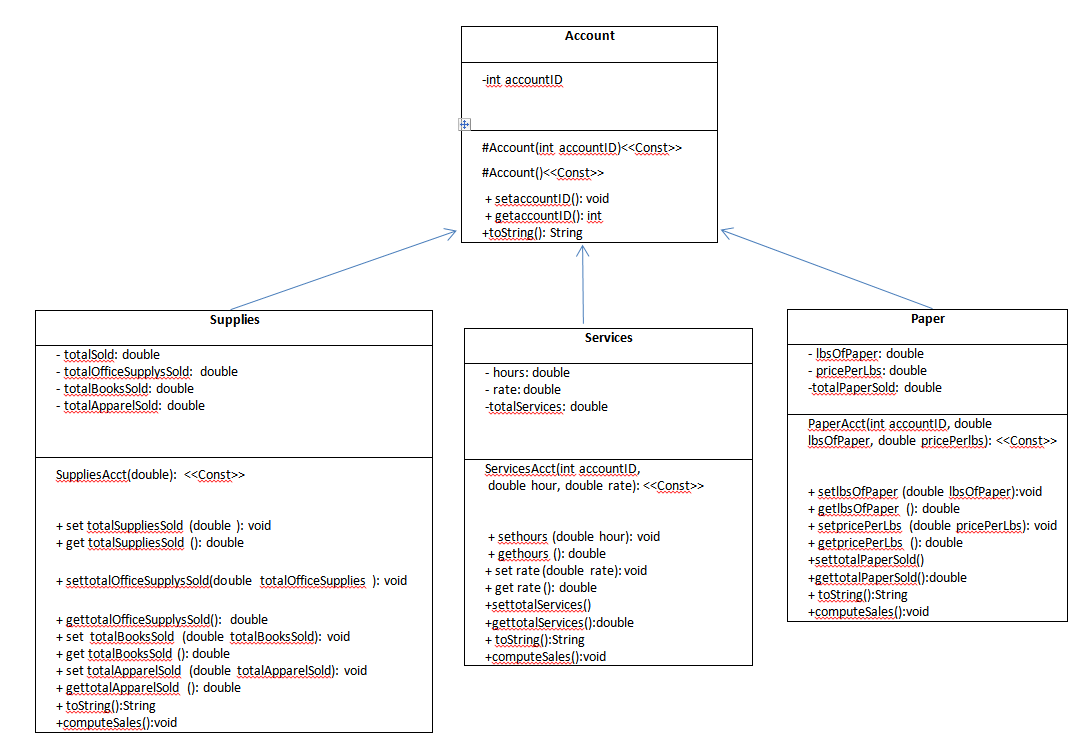


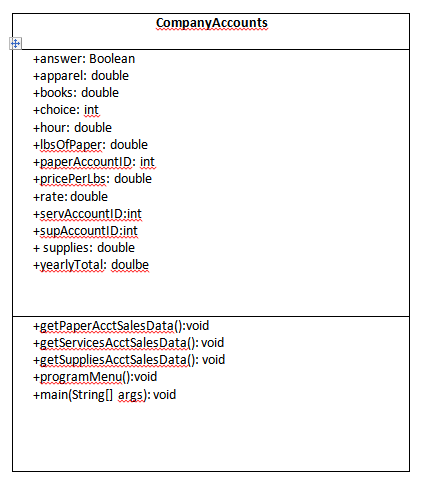
# Phase 5 Final

The following program allows the company S & S’s sale employees (SA) to enter sales data for the company’s three account types, Supplies, Services, and Paper accounts. The program will display a welcome message and a menu with five choices. Choice 1-3 will allow the SA to enter all sales data. At any given time the SA may select option 4 to calculate and display the totals for each account as well as the yearly total. Option 5 allows the user to exit the program.

The following are the class diagram, pseudocode, and program screenshots captured at runtime.

## UML Class Diagram:





## Pseudocode:

Main(){

Instantiate 3 account types

Display welcome

Try{

While{

Display program menu

User inputs option number

Switch{

Case 1 getSuppliesAcctSalesData()

Case 2 getServicesAcctSalesData()

Case 3 getPaperAcctSalesData()

Case4 instatiate array with account types

For loop computes and displays account totals

Yearly total is calculated and displayed

Displays the frame with the logo and yearly total

Case 5 terminates the program

Default if the user enters a number other than in the range of 1-5

It will display an error message

}//end switch

}//end while

}//end try

Catch {

Catches the InputMismatchExeption if the user enters anything other than a number

Displays error message and Terminates the program

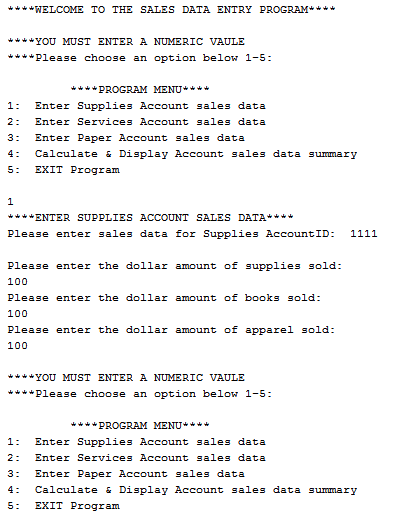
}//end catch

}//end main()

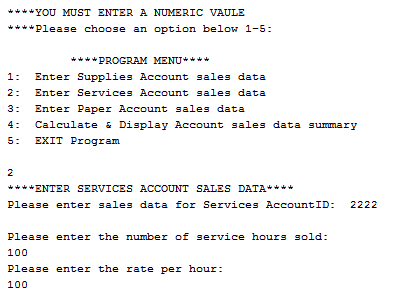
## Runtime Screenshots:

Welcomes the user

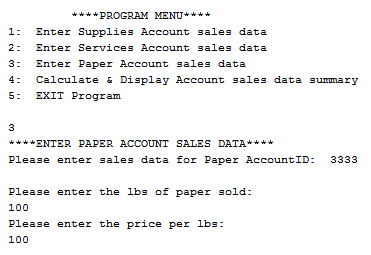
User enters option 1 and enters values



User chooses option 2 and enters values

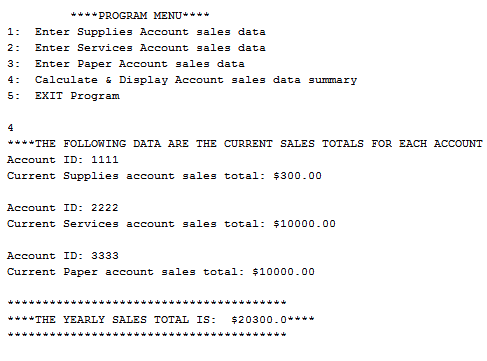


User enters option 3 and enters values

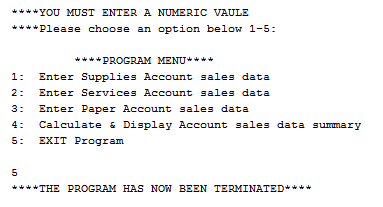


User enters option 4

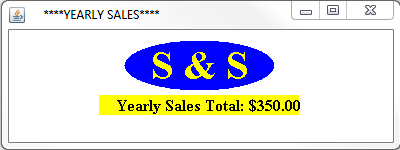
Program calculates account totals and displays yearly total



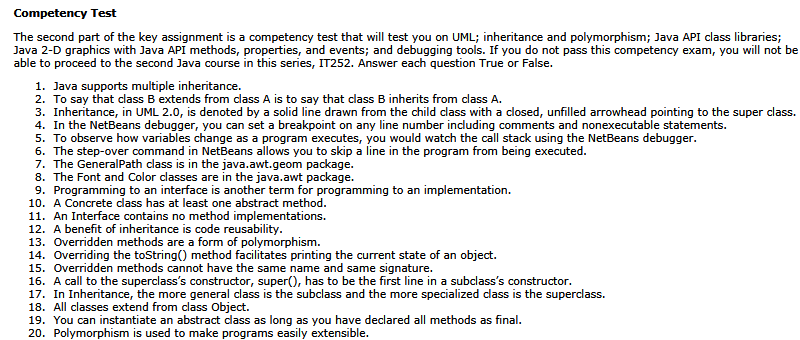
User enters option 5 and the program terminates



Each time option 4 is selected the logo is displayed with the yearly total



# Competency Test



1. T
2. T
3. T
4. F
5. F
6. F
7. T
8. T
9. F
10. F
11. T
12. T
13. T
14. T
15. F
16. T
17. F
18. T
19. F
20. T

# References

Abstract Methods and Classes (The Java™ Tutorials > Learning the Java Language > Interfaces and Inheritance). (n.d.). Retrieved from http://docs.oracle.com/javase/tutorial/java/IandI/abstract.html

Java Inheritance. (n.d.). Retrieved from <http://www.javabeginner.com/learn-java/java-inheritance>

The UML Class Diagram: Part 1 - Developer.com - Developer.com. (n.d.). Retrieved from <http://www.developer.com/design/article.php/2206791/The-UML-Class-Diagram-Part-1.htm>