CS116

LAB 4

Lab 4 is due on Friday March 07 on Blackboard by 10:00 p.m time stamped. Use EditPlus or equivalent text editor to create source code.

This Lab. is worth 3 points

Objectives:

1. Designing an algorithm for data formatting (working with algorithms).
2. Multidimentional Arrays and formatting displays.
3. Working with ArrayList.
4. Multiple service and helper classes.
5. Compiling classes that have cross referencing.
6. Inheritance.
7. Abstract classes.
8. Polymorphism.

Task 1 (1.5 points)

In this task we are going to use classes provided to you on which you will need to add your own code. You can download those classes from the course's web site by going to the date that Lab 4 was posted (link: HelpClassesForLab4Task1.zip). These class files provided are:
Enumeration classes: Dealer.java and VehicleEngine.java
Service classes: Vehicle.java and CarDealer.java
Client class: VehicleClient.java (incomplete class)

Study the code provided and make the changes requested in this Task. Make sure that the work done on this task 1 is on its own folder named Task 1 (do not mix Task 1 and Task 2 files together). The text data file is also provided to you.

Leave the packages as they are in the provided code. The following grading rules apply

1. **Any changes to the provided package names will cost you 0.5 points (including using the same names from a previous year solution)!!!!!!!**

2. You must compile the files provided to you before you make any of the changes requested in this lab. There are two service classes: Vehicle.java and CarDealer.java. A technique called object cross referencing is used in those two classes. This means that there is an attribute of type CarDealer in the Vehicle class and an attribute Vehicle in the CarDealer class. Therefore in trying to compile we have an issue- which comes first. Therefore to compile:

   First compile the two enum classes
   To compile the two service classes : first comment out any references to CarDealer (attribute, accessor/mutator and toString) in the Vehicle class.Then compile the Vehicle class.
   Now go ahead and compile the CarDealer class.
   Go back to the Vehicle class and get rid of the commenting out (now CarDealer referencing is O.K.). Recompile the Vehicle class. Not recompiling the classes will make the rest of your code changes useless and you will cause you to lose 1 point.

- The new task is to create a display of sales of vehicles sold, in the form of a 3 dimensional table. Sold vehicles will be tracked by dealership, week number and day of the week. There are 6 days in a week that dealerships are open Monday, Tuesday, Wednesday, Thursday, Friday and Saturday. A dealer can sell another dealer's vehicle, therefore who has possession of the car (as per the data of the text file) does not make any difference as far as the sale of the car is concerned.
  - A three dimensional array will hold the sale price of each vehicle sold.
- <u>Vehicles are sold as follows:</u>
  The 5 dealers have sold the below listed vehicles by id number. The id numbers correspond to the object id numbers created by the Vehicle class when Vehicle objects are created based on the text file's data. The pattern of the sales by dealer (dealer's name below is the name of the dealer who sold the car and not the dealer who had possession of the car) is as follows:

Week #1	Week #2
<u>ID1:</u>	<u>ID1:</u>
Monday : vehicle id: #1 is sold (its price is the data)	Monday : vehicle id: #16 is sold (its price is the data)
Wednesday: vehicle id: #2 is sold (its price is the data)	Wednesday: vehicle id: #17 is sold (its price is the data)
Friday: vehicle id: # 3 is sold (its price is the data)	Friday: vehicle id: # 18 is sold (its price is the data)
<u>ID2:</u>	<u>ID2:</u>

Tuesday: vehicle id: # 4 is sold (its price is the data)	Tuesday: vehicle id: # 19 is sold (its price is the data)
Thursday: vehicle id:# 5 is sold (its price is the data)	Thursday: vehicle id: # 20 is sold (its price is the data)
Saturday: vehicle id: # 6 is sold (its price is the data)	Saturday: vehicle id:# 21 is sold (its price is the data)
ID3:	ID3:
Monday: vehicle id: #7 is sold (its price is the data)	Monday: vehicle id: #22 is sold (its price is the data)
Wednesday: vehicle id: #8 is sold (its price is the data)	Wednesday: vehicle id: #23 is sold (its price is the data)
Friday: vehicle id: #9 is sold (its price is the data)	Friday: vehicle id: #24 is sold (its price is the data)
ID4:	ID4:
Tuesday: vehicle id: # 10 is sold (its price is the data)	Tuesday: vehicle id: # 25 is sold (its price is the data)
Thursday: vehicle id: #11 is sold (its price is the data)	Thursday: vehicle id: #26 is sold (its price is the data)
Saturday: vehicle id: #12 is sold (its price is the data)	Saturday: vehicle id: #27 is sold (its price is the data)
ID5:	ID5:
Monday: vehicle id: #13 is sold (its price is the data)	Monday: vehicle id: #28 is sold (its price is the data)
Wednesday: vehicle id: #14 is sold (its price is the data)	Wednesday: vehicle id: #29 is sold (its price is the data)
Friday: vehicle id: #15 is sold (its price is the data)	Friday: vehicle id: #30 is sold (its price is the data)

Based on the above table you will have to come up with an algorithm to store data in a 3 - dimensional array. The data of course is made out of the sale price of Vehicle objects. The dimensions of the array are: 1st dimension: Dealer, 2nd dimension: week , 3rd dimension :day Study the pattern of sales in order to develop your algorithm.

- Since we need data for 30 vehicles, a text file with data is provided:.

  1.5,2,owner1,Type2,ID1,17900.20
  2.0,3,owner2,Type2,ID3,10000.0
  3.3,4,owner3,Type1,ID2,20000.0
  4.0,5,owner4,Type4,ID5,45000.30
  5.3,6,owner5,Type4,ID1,23050.0
  2.5,4,owner6,Type4,ID2,10000.40
  5.3,3,owner7,Type3,ID2,23040.0
  3.0,3,owner8,Type3,ID5,34000.30
  2.4,5,owner9,Type1,ID3,45000.0
  4.0,2,owner10,TYpe4,ID4,22000.0
  2.0,8,owner11,Type4,ID1,35000.20

1.0,3,owner12,Type1,ID5,16000.0
2.9,1,owner13,Type3,ID3,24500.00
4.0,2,owner14,Type3,ID1,12750.25
8.0,4,owner15,Type2,ID1,19600.00
1.0,1,owner16,Type2,ID2,25000.00
2.0,2,owner17,Type2,ID5,20000.50
6.0,1,owner18,Type1,ID2,34000.00
9.0,1,owner19,Type2,ID1,56000.00
9.0,2,owner20,Type3,ID1,19000.40
3.0,8,owner21,Type4,ID4,37000.00
4.0,4,owner22,Type4,ID5,28000.00
5.0,1,owner23,Type1,ID3,50000.00
4.0,3,owner24,Type2,ID2,44000.00
3.5,4,owner25,Type3,ID2,45600.00
1.8,1,owner26,Type1,ID5,15000.00
2.0,3,owner27,Type4,ID1,14000.03
3.0,4,owner28,Type3,ID3,18000.0
2.7,3,owner29,Type1,ID4,24000.00
4.0,3,owner30,Type2,ID5,10000.00

Note: Make sure that there is no space between characters in the text file.

- When a vehicle is sold it needs to be marked as sold. Therefore add an instance variable in the ***Vehicle class*** which is responsible for marking a vehicle "sold". You need to figure the data type of this variable. Initialize the variable in both constructors (initialization value should show the vehicle as not sold). **The arguments of the non default constructor do not change**. Add accessor/mutator methods for the new instance variable and also add the variable in the toString method with a short descriptive String ahead of it i.e "Sold status="+..

- In the client class called VehicleClient make the following changes.

    - Create a NON static method called **tripleArray** which takes as argument the ArrayList of Vehicle objects and returns a triple array of double values that represent the prices of the cars sold. The double values represent the sales for each day of the week for each dealer. The first dimension is the dealer

      Index 0: ID1
      Index 1: ID2
      Index 2: ID3
      Index 3: ID4
      Index 4: ID5

The second dimension is the weeks.
Index 0: week 1
Index 1: week 2

The third dimension are the 6 days (index 0: Monday- index 5: Saturday)

Every time a price is set in the array the Vehicle 's sale status is changed into sold.

Create the algorithm that would allow you to enter into the proper cell of the 3 dimensional table , represented by the 3-dim. array, the total sales for each of the days for each of the weeks from all dealers for that day. THE ALGORITHM SHOULD UTILIZE for LOOPS. DO NOT INSERT THE OBJECTS ONE AT A TIME USING 30 LINES OF CODE **(-1.0 points if you do that)!!! Remember that this task is not so much about programming techniques as it is about structuring an algorithm.**

THEORY

Three for loops are needed to establish the algorithm. Let the first for loop iterator i represent the dealers (indices 0-4), the second for loop iterator j represents the weeks (indices 0 and 1) and the third for loop iterator h represents the days of the week (indices 0 to 5).

i.e. in the sample code below the counters are used to find the proper Vehicle id for the each dealer:

```
for(int i=0; i<=sales.length-1; i++)
{
    for(int j=0; j<=sales[i].length-1; j++)
    {
            for(int h=0; h<=sales[i][j].length-1; h++)
            {
                    //if oneDealer and day is M,W,or F
                    if((i==0)&&(h==0||h==2||h==4))
                    {

                       if(j==0){
                         sales[i][j][h]=sales[i][j][h]+al.get(countw1).getCarDealer().getSale();
                         al.get(countw1).setSold(true);
                         countw1++;
                         countw2=countw1;
                        }
                        else{
                         sales[i][j][h]=sales[i][j][h]+al.get(countw2+12).getCarDealer().getSale();
```

```
                    al.get(countw2+12).setSold(true);
                     countw2++;
                }
```

……

……

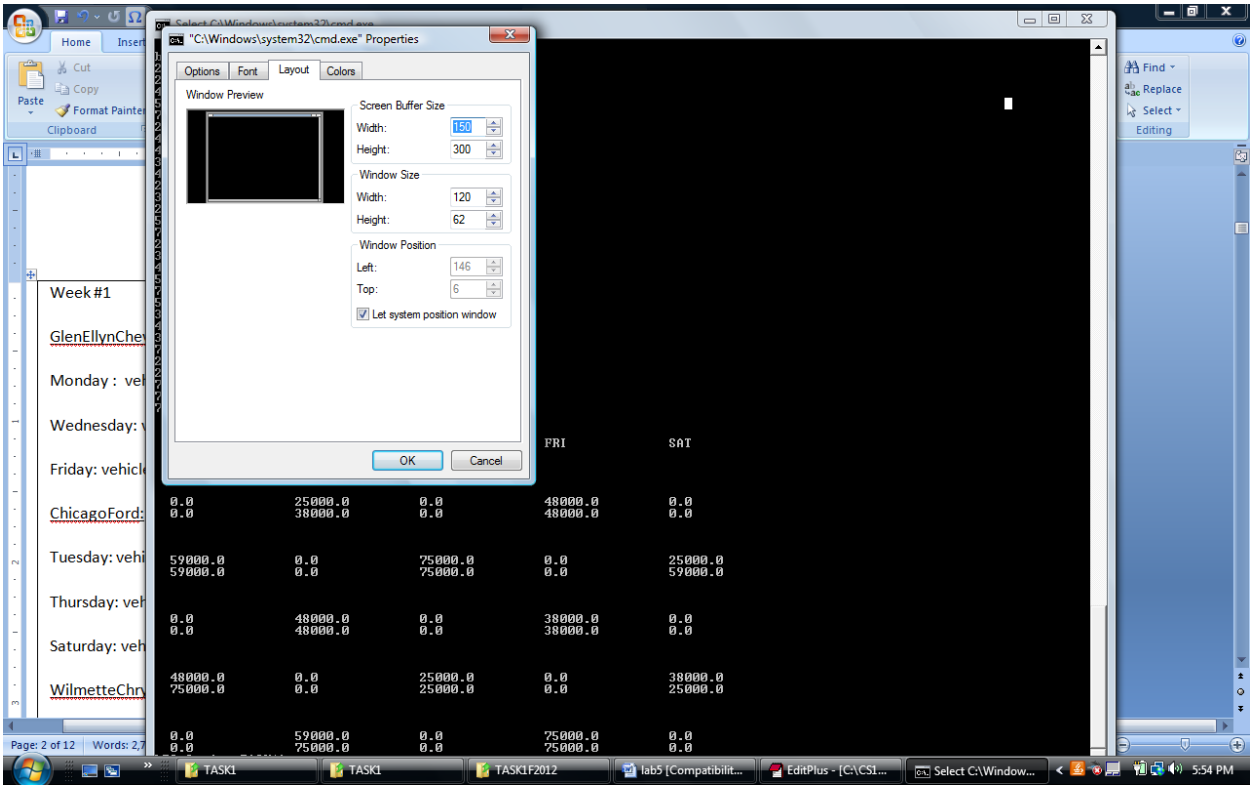
- In the main method:

    - Create an ArrayList of Vehicle objects.
    - Create the array for CarDealer objects.
    - Set the CarDealer attribute of each Vehicle in the array list to a corresponding CarDealer object from the array of CarDealer objects.
    - Output (display) the price of each vehicle in the ArrayList together with the vehicle id and the sale status (at this time nothing is sold). <u>You must use the new for loop format for ArrayList. Call it : FIRST OUTPUT</u>
    - Call the method tripleArray and receive the 3 dimensional array . Next output (display) the sales per day from the 3 dimensional array. Format the output so that it appears as shown below in the SAMPLE OUTPUT shown below, formatted as a table. Use escape characters \n for new line and \t for tab as required. Remember that there is a command **<u>System.out.print</u>** that does not enter a new line when it prints. Make sure the headers appear as shown and in the proper location of the table. Call it SECOND OUTPUT.
      THIRD OUTPUT is the iteration through the array list one more time (this time the sale status should be: sold) displaying just the id of each vehicle and its sale status.

<u>Note 1:</u> You may not be able to observe the table correctly with the present size of your DOS window. As a result you could change the size of the window. To do that :

- Click in the upper left corner of the DOS window and choose properties.
- In the properties window choose Layout.
- In the Layout window advance the screen width to i.e 120.

<u>Note 2:</u> You will need three for loops (one inside the other) but certain logic needs to be implemented inside the for loops both in the case of building the 3-dim array in the method tripleArray and also in formatting the display.



Compile the classes and run the program

Sample Output

```
C:\CS116\SPRING2014\Labs\Lab4\Lab4Solution\TASK1>java folder1.VehicleClient
----------FIRST OUTPUT ----------
The price of car with id= 1 is : 17900.2 The sale status is: false
The price of car with id= 2 is : 10000.0 The sale status is: false
The price of car with id= 3 is : 20000.0 The sale status is: false
The price of car with id= 4 is : 45000.3 The sale status is: false
The price of car with id= 5 is : 23050.0 The sale status is: false
The price of car with id= 6 is : 10000.4 The sale status is: false
The price of car with id= 7 is : 23040.0 The sale status is: false
The price of car with id= 8 is : 34000.3 The sale status is: false
The price of car with id= 9 is : 45000.0 The sale status is: false
The price of car with id= 10 is : 22000.0 The sale status is: false
The price of car with id= 11 is : 35000.2 The sale status is: false
The price of car with id= 12 is : 16000.0 The sale status is: false
The price of car with id= 13 is : 24500.0 The sale status is: false
The price of car with id= 14 is : 12750.25 The sale status is: false
The price of car with id= 15 is : 19600.0 The sale status is: false
The price of car with id= 16 is : 25000.0 The sale status is: false
The price of car with id= 17 is : 20000.5 The sale status is: false
```

The price of car with id= 18 is : 34000.0 The sale status is: false
The price of car with id= 19 is : 56000.0 The sale status is: false
The price of car with id= 20 is : 19000.4 The sale status is: false
The price of car with id= 21 is : 37000.0 The sale status is: false
The price of car with id= 22 is : 28000.0 The sale status is: false
The price of car with id= 23 is : 50000.0 The sale status is: false
The price of car with id= 24 is : 44000.0 The sale status is: false
The price of car with id= 25 is : 45600.0 The sale status is: false
The price of car with id= 26 is : 15000.0 The sale status is: false
The price of car with id= 27 is : 14000.03 The sale status is: false
The price of car with id= 28 is : 18000.0 The sale status is: false
The price of car with id= 29 is : 24000.0 The sale status is: false
The price of car with id= 30 is : 10000.0 The sale status is: false
----------SECOND OUTPUT----------

	MON	TUE	WED	THU	FRI	SAT
ID1						
week 1:	**17900.2**	**0.0**	**10000.0**	**0.0**	**20000.0**	**0.0**
week 2:	**25000.0**	**0.0**	**20000.5**	**0.0**	**34000.0**	**0.0**
ID2						
week 1:	**0.0**	**45000.3**	**0.0**	**23050.0**	**0.0**	**10000.4**
week 2:	**0.0**	**56000.0**	**0.0**	**19000.4**	**0.0**	**37000.0**
ID3						
week 1:	**23040.0**	**0.0**	**34000.3**	**0.0**	**45000.0**	**0.0**
week 2:	**28000.0**	**0.0**	**50000.0**	**0.0**	**44000.0**	**0.0**
ID4						
week 1:	**0.0**	**22000.0**	**0.0**	**35000.2**	**0.0**	**16000.0**
week 2:	**0.0**	**45600.0**	**0.0**	**15000.0**	**0.0**	**14000.03**
ID5						
week 1:	**24500.0**	**0.0**	**12750.25**	**0.0**	**19600.0**	**0.0**
week 2:	**18000.0**	**0.0**	**24000.0**	**0.0**	**10000.0**	**0.0**

----------THIRD OUTPUT----------

The price of car with id= 1 The sale status is: true
The price of car with id= 2 The sale status is: true
The price of car with id= 3 The sale status is: true
The price of car with id= 4 The sale status is: true
The price of car with id= 5 The sale status is: true
The price of car with id= 6 The sale status is: true
The price of car with id= 7 The sale status is: true
The price of car with id= 8 The sale status is: true
The price of car with id= 9 The sale status is: true
The price of car with id= 10 The sale status is: true
The price of car with id= 11 The sale status is: true
The price of car with id= 12 The sale status is: true
The price of car with id= 13 The sale status is: true
The price of car with id= 14 The sale status is: true
The price of car with id= 15 The sale status is: true
The price of car with id= 16 The sale status is: true
The price of car with id= 17 The sale status is: true
The price of car with id= 18 The sale status is: true
The price of car with id= 19 The sale status is: true
The price of car with id= 20 The sale status is: true
The price of car with id= 21 The sale status is: true
The price of car with id= 22 The sale status is: true
The price of car with id= 23 The sale status is: true
The price of car with id= 24 The sale status is: true
The price of car with id= 25 The sale status is: true
The price of car with id= 26 The sale status is: true
The price of car with id= 27 The sale status is: true
The price of car with id= 28 The sale status is: true
The price of car with id= 29 The sale status is: true
The price of car with id= 30 The sale status is: true

Task 2 (1.5 points)

The following grading rules apply: Any names of attributes or methods or code that matches a previous solution or an equivalent lab , will cost you -1.0 point!!

In this task we are going to build an inheritance relationship with polymorphism.

The classes described need to be packaged as described.

1. Create an Enumeration called Jobs with data ELECTRICAL_ENGINEER, MECHANICAL_ENGINEER, ADMINISTRATIVE _SECRETARY, ADMINISTRATIVE_ASSISTANT, ENGINEERING_MANAGER, ADMINISTRATIVE_MANAGER, NONE. This enum clss resides in package Client.Services.Enums.

2. Create a class called Worker in package Client.Services. This class is ***abstract***. It has attributes:

   public String name;

   public int socialSecurity;

   private int yearsExperience;

   public Jobs et;

   public static int id;

   public int currentID;

   a. Create a default constructor that initializes the attributes to : AnyName, 12345, 0, NONE advances **id** every time it is called , and assigns it to currentID.

   b. Create a non default (overloaded) constructor that has as arguments identifiers for the attributes: name , socialSecurity ,yearsExperience, Jobs. It also advances the id every time it is called and assigns it to currentID.

   c. Produce the proper accessor, mutator (for all attributes) and toString methods.

   d. The class Worker, in addition, has an **abstract** method called ***benefitsCalculation*** which takes as argument an enumeration Jobs and returns a double that represents the benefits for the type of employee job.

   e. Compile the class Worker.

3. Produce a class called ***Engineer*** which **inherits Worker** class and resides in package Client.Services. This class has no main method. It has the specialized attributes: ***private double weeklyBenefits.***

   a. This class also has a non default (overloaded) constructor which accepts the same 4 arguments as the super class, and in addition the argument for the specialized variable weeklyBenefits. It calls the super constructor and passes the 4 arguments that it needs and then calls the mutator methods for weeklyBenefits to set the values of the attributes to the corresponding arguments.

   b. It has a default constructor which calls the super class' constructor and also sets the specialized variable of this class to 400.00.

   c. It has an implementation of abstract method ***benefitsCalculation.*** This method calculates the benefits as follows:

If the Engineer is ELECTRICAL_ENGINEER then

benefits= weeklyBenefits+yearsExperience*80.00;

if the Enginneer is MECHANICAL_ENGINEER then

benefits= weeklyBenefits/2+yearsExperience*120.00;

Any other Job type set for an Engineer object will cause the benefits to be zero.

It returns benefits as a double.

d. It has accessor and mutator methods for its specialized attributes and also a toString method. The toString method calls the super toString and then adds to it the value of the additional instance variables declared in this class.

e. Compile this class.

4. Create a similar class named ***AdministrativePersonnel*** which also inherits Worker and resides in package Client.Services***.*** This class has the specialized attributes: ***private double rate, private double hours.***

It has No main method.

a. It has default and non default constructors similar to the previous class (where the super class constructor is called first and the additional specialized parameters are also set by calling their mutator methods). The non default constructor takes 6 arguments (4 for the super and 2 for this class). The default constructor sets the specialized variables to 10.0, 10.0 respectively.

b. It implements the method ***benefitsCalculation*** based on the formula:

If the employee is ADMINISTRATIVE_SECRETARY then

benefits=rate*hours+yearsExperience*15.00;

if the employee is ADMINISTRATIVE_ASSISTANT

benefits=rate*hours+yearsExperience*25.00

Any other Job type set for an AdministrativePersonnel object will cause the benefits to be zero.

it returns the benefits as a double.

c. It also has accessor and mutator methods for its specialized attributes as well as toString method, similar as in the previous class Engineer.

d. Compile this class.

5. Create another class named ***Management*** which also inherits Worker and resides in package Client.Services (and no main method)***.*** It has specialized attributes: ***private double weeklyBenefits, private double bonus*****.** Set up the default and non default constructors like in the previous two classes (make sure that you have the correct number of arguments in the non default constructor). Add the accessor and mutator methods for the specialized instance variables and the toString method as before.

a. It also implements the method ***benefitsCalculation*** based on the formula:

A Manager who is ENGINEERING_MANAGER is compensated using:

benefit=weeklyBenefit+bonus

A Manager who is ADMINISTRATIVE_MANAGERis compensated using:

benefit=weeklyBenefit+0.5*bonus

Any other Job type set for a Manager object will cause the benefit to be zero.

It returns benefit as a double.

b. Compile this class.

Task 2 continued

1. Create a class called WorkerBenefits . This class resides in package Client. This class has the methods describe below and acts as a client to the hierarchical tree of Worker classes created. Notice that there is no inheritance here. Class WorkerBenefits "uses" classes Engineer, AdministativePersonnel, Management and their superclass Worker.

2. Create a method called listOfWorkers.

   public ArrayList<Worker> listOfWorkers()

   This method takes no arguments and returns an ArrayList of Worker type. This method creates objects of each type of the 3 workers we have (Engineer, AdministrativePersonnel, Management) by reading the data from the file workers.txt. The file is given as part of this lab. There is one line of data for each worker in the text file. The tokens are separated by

comas. Your code should recognize the type of worker from the first String in a line of data. The next tokens in the line are the data for that type of Worker. For example;
ELECTRICAL_ENGINEER, John Doe, 1234, 10,1000.00
The first token identifies the type of Job (based on that type an Engineer object is needed), the next token is the name of the engineer, the next token is the social security, the next token is the years of experience and the last is the weekly benefits (the values and data types corresponding to the values needed by the constructor of Worker class). The same applies to other lines of data in the text file (values are given as needed by the corresponding constructor of the other types of workers) i.e

ADMINISTRATIVE_SECRETARY,Helen Adm1,45123,10,20.0,40.0
Type of Job, name of employee, social security num, years of experience, rate, hours worked.
ADMINISTRATIVE_MANAGER,Ann Man4,62345,10,2300.00,400.00
Type of Job, name of employee, social security num, years of experience, weekly benefit, bonus.

For each line read, create the object that relates to that type of Worker (Engineer, AdministrativePersonnel, Management objects).
Store the objects created into an ArrayList of type <Worker> and return it.

FILE worker.txt data:

ELECTRICAL_ENGINEER,John Eng1,12345,10,1000.00
ENGINEERING_MANAGER,Jim Man1,23451,10,1500.00,500.00
ADMINISTRATIVE_MANAGER,Jim Man2,34512,2, 2000.00,100.00
ADMINISTRATIVE_SECRETARY,Helen Adm1,45123,10,20.0,40.0
MECHANICAL_ENGINEER,Maria Eng2,51234,12,1200.00
ENGINEERING_MANAGER,Jim Man3,23456,3,1200.00,600.00
ADMINISTRATIVE_ASSISTANT,John Doe1,34562,20,15.0,30.0
ELECTRICAL_ENGINEER,Mary Eng3,45612,20,2000.00
ADMINISTRATIVE_ASSISTANT,Nick Doe2,56234,5,22.0,45.0
ADMINISTRATIVE_MANAGER,Ann Man4,62345,4,2300.00,400.00

3. Add a method called displayData.

public void displayData(ArrayList<Worker> ale)

It returns void and takes as argument the ArrayList of Worker objects. In this method, capture the benefits for each type of employee stored in the array list. Output the benefits paid to each worker, including the job type, the employees name and object ID number using System.out..
You should call the corresponding benefitsCalculation method using polymorphism (read the lecture on polymorphism and the corresponding chapter in the text).

- Make sure that you create the proper reference for Worker and use it to invoke the benefitsCalculation method for each Worker object in the array list.
- You can first iterate through the list by using the new ***for loop*** format for iterating through an ArrayList, each iteration calculating the benefits and capturing the returned amount and outputting it.

4. Add a main method. In the main method retrieve the ArrayList of Worker objects by calling the method listOfWorkers and then call the display method displayData to display employee data.
5. Compile this class an run the program.

SAMPLE OUTPUT

---------- Run Java ----------

```
C:\CS116\SPRING2014\Labs\Lab4\Lab4Solution\TASK2>java Client.WorkerBenefits
The benefit is 1800.0 The name is: John Eng1  The Job type type is ELECTRICAL_ENGINEER The id is: 1
The benefit is 2000.0 The name is: Jim Man1  The Job type type is ENGINEERING_MANAGER The id is: 2
The benefit is 2050.0 The name is: Jim Man2  The Job type type is ADMINISTRATIVE_MANAGER The id is: 3
The benefit is 950.0 The name is: Helen Adm1  The Job type type is ADMINISTRATIVE_SECRETARY The id is: 4
The benefit is 2040.0 The name is: Maria Eng2  The Job type type is MECHANICAL_ENGINEER The id is: 5
The benefit is 1800.0 The name is: Jim Man3  The Job type type is ENGINEERING_MANAGER The id is: 6
The benefit is 950.0 The name is: John Doe1  The Job type type is ADMINISTRATIVE_ASSISTANT The id is: 7
The benefit is 3600.0 The name is: Mary Eng3  The Job type type is ELECTRICAL_ENGINEER The id is: 8
The benefit is 1115.0 The name is: Nick Doe2  The Job type type is ADMINISTRATIVE_ASSISTANT The id is: 9
The benefit is 2500.0 The name is: Ann Man4  The Job type type is ADMINISTRATIVE_MANAGER The id is:
```

10 Submission instructions

- In your submission you must include
  a. This document with the answers (copies of the outputs as requested above).
  b. The source code files and the compiled files (the package) from Task 1 in Task1 folder.
  c. The source code files and the compiled files (the package) required for Task 2 in the Task 2 folder.

- Zip all files and name the zip file using your first name followed by your last name followed by lab1.

  i.e. George_KayLab4.zip
- Upload the file on Blackboard in assignment folder “Lab 5”.