**CPSC165**

**Assignment#2**

**30 Points**

**PART I.**

**Understand the UML Class Diagram**

Use the following UML diagram to build the class. The first section specifies the attributes. The second section specifies the behaviors, and the first character specifies the access modifier value, where:

* "-" means that the class member is private, and
* "+" means that the class member is public.



**Code the Employee Class**

1. Using the provided Class Diagram from Step 1, code the Employee class.
2. The default constructor should set the attributes as follows: firstName = "not given", lastName = "not given", gender = "U" (for unknown), dependents = 0, and annualSalary = 20,000.
3. The multi-arg constructor should initialize all of the attributes using values passed in using its parameter list.
4. As shown in the Class diagram, each attribute should have a "getter" to retrieve the stored attribute value, and a "setter" that modifies the value.
5. The calculatePay( ) method of the Employee class should return the value of annual salary divided by 52 (return annualSalary / 52;).
6. The displayEmployee() method should display all the attributes of the Employee object in a well-formatted string with logical labels applied to each attribute. Don't forget to call calculatePay from within the displayEmployee method in order to display the Employee's weekly pay as well!

**Code the Main Program**

In the Main class, create code statements that perform the following operations. Be sure you follow proper commenting and programming styles (header, indentation, line spacing, etc.).

1. Create an Employee object using the default constructor.
2. Prompt for and then set the first name, last name, gender, dependents, and annual salary. **(Remember that you have to convert gender, dependents, and annual salary from strings to the appropriate data type.)**
3. Display a divider that contains the string "Employee Information". Write a function to do this.
4. Format the currency.
5. Display the Employee information.
6. Create a second Employee object using the multi-argument constructor, setting each of the attributes with appropriate valid values.
7. Display a divider that contains the string "Employee Information". Use the function from line 3 do this.
8. Display the Employee information for the second Employee object.

**Compile and Test**

When done, compile and run your code.

Then, debug any errors until your code is error-free.

Check your output to ensure that you have the desired output, modify your code as necessary, and rebuild.

**Submit Deliverables**

* Capture the Console output window and paste into a Word document. Zip the output file along with the source code files.
* Upload to canvas.

**PART II.**

**Update the Employee Class**

* add the following attribute:
	+ static numEmployees: int = 0
* add the following behaviors:

+ static getNumEmployees( ) : int

+ setDependents(in dep : String) : void

+ setAnnualSalary(in sal : String) : void

### Modify the Employee

1. Code the changes to the Employee class.
	1. Create a static numEmployees variable and initialize it to zero.
	2. Increment numEmployees by 1 in each of the constructors.
	3. Create an overloaded setDependents method and this time make the parameter a string.
	4. Create an overloaded setAnnualSalary method and this time make the parameter a string.
	**Remember that you will have to convert the string in the above two "set" methods to the data type of the attribute.**
	5. Make the getNumEmployees a static method. (This way, you can call it with the class name instead of an object name.)

Be sure you follow proper commenting and programming styles (indentation, line spacing, etc.).

### Modify the Main Method

In the Main class, create code statements that perform the following operations. Be sure you follow proper commenting and programming styles (header, indentation, line spacing, etc.). Note that several of the steps below were accomplished in part I. New steps are in **bold**.

1. Create an Employee object using the default constructor.
2. Prompt for and then set the first name, last name, and gender. Consider using your getInput method from Week 1 to obtain data from the user for this step as well as Step 3.
3. **Prompt for and then set dependents and annual salary using the new overloaded setters.**
4. Using your code from Week 1, display a divider that contains the string "Employee Information".
5. Display the Employee Information.
6. **Display the number of employees created using getNumEmployees. Remember to access getNumEmployees using the class name, not the Employee object.**
7. Create a second Employee object using the multi-arg constructor, setting each of the attributes with the following values: **"Mary", "Noia", 'F', 5, 24000.0**
8. Using your code from Week 1, display a divider that contains the string "Employee Information".
9. Display the employee information for the second Employee object.
10. **Display the number of employees created using getNumEmployees. Remember to access getNumEmployees using the class name, not the Employee object.**

### Compile and Test

When done, compile and run your code.

Then, debug any errors until your code is error-free.

Check your output to ensure that you have the desired output, modify your code as necessary, and rebuild.

### Submit Deliverables

* Capture the Console output window and paste into a Word document. Zip the output file along with the source code files.
* Upload to canvas.